Habilitation à Diriger les Recherches

Spécialité Informatique

présentée à

l'Université de Strasbourg

par

David CAZIER

Structures combinatoires pour la modélisation, l'animation et le traitement de la géométrie

---

13 décembre 2010

M. Jean-François Dufourd, garant,
professeur à l'université de Strasbourg.

M. Luc Brun, rapporteur,
professeur à l'ENSICAEN.

M. Bruno Lévy, rapporteur,
directeur de recherche INRIA à Nancy.

M. Christophe Schlick, rapporteur,
professeur à l'université de Bordeaux.

M. Philippe Meseure, examinateur,
professeur à l'université de Poitiers.

Habilitation préparée au "*Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection*", dans l'équipe de recherche "*Informatique Géométrique et Graphique*".

Remerciements

La rédaction d'une habilitation à diriger les recherches est une étape importante dans la carrière d'un enseignant chercheur. Pendant dix ans au sein de l'équipe d'informatique graphique, j'ai bénéficié d'un environnement de travail cordial et stimulant, ce dont je dois remercier tous ses membres et en particulier Dominique Bechmann qui m'a encouragé à franchir le pas et soutenu durant toutes ces années.

Je tiens à remercier Jean-François Dufourd qui avait déjà dirigé ma thèse et qui m'a à nouveau accompagné dans ce travail. Je souhaite également remercier Luc Brun, Bruno Lévy, Christophe Schlick et Philippe Meseure qui ont accepté de relire et juger mon travail.

Je veux enfin remercier Pierre Kaemer, Thomas Jund, Cyril Kern et Sylvain Thery avec qui j'ai travaillé sur tous ces projets. Ils ont porté avec moi une partie des travaux présentés dans ce mémoire. Les nombreuses discussions que nous avons eues ont été une source d'inspiration précieuse.

à Marie,

Table des matières

Chapitre 1

Introduction

Depuis l'essor de l'informatique graphique et de la vision par ordinateur dans les années 80, un grand nombre de modèles ont été proposés. Ils permettent de décrire et manipuler des objets géométriques de dimensions et de structures variées dont l'implantation est souvent liée aux applications visées. Les traitements numériques et graphiques que l'on peut effectuer sur de tels objets dépendent pour beaucoup du choix d'une représentation ou d'un modèle. Ce choix détermine le domaine de représentation des objets construits et manipulés ainsi que les formes, les structures, les propriétés physiques ou visuelles qui pourront leur être associées. Le modèle utilisé influence également les possibilités logicielles en termes d'interaction, d'édition ou d'efficacité des traitements. En particulier, la qualité, la robustesse et la complexité des algorithmes géométriques ou graphiques sont grandement influencées par le choix d'une représentation.

De manière très générale un objet est constitué d'un ensemble de points d'un espace de plongement. Ces ensembles de points doivent être discrétisés pour pouvoir être manipulés par un programme. Suivant les applications, ils sont subdivisés en des ensembles finis de primitives ou de cellules, plus ou moins structurés. Il existe différentes familles de discrétisation, possédant des caractéristiques propres en termes de coût mémoire et de temps d'accès aux informations stockées. Elles sont souvent adaptées à certains types de problèmes et supportent efficacement une gamme plus ou moins large de traitements.

Les modèles les plus répandus sont les modèles de représentation par les bords. Ils ont d'abord été conçus pour la manipulation de polyèdres ou de surfaces, comme dans les travaux de Baumgart [6], Requicha [117], Mäntilä [107], Guibas et Stolfi [64] ou Weiler [133, 132]. Ils ont été étendus aux décompositions cellulaires de dimensions supérieures et notamment volumiques, par exemple par Dobkin et Laszlo [46] ou Ansaldi et De Floriani [2]. Dans ce type de représentation chaque objet est découpé ou subdivisé en un ensemble fini de cellules, de formes et de natures variables, équipées de relations d'incidence ou d'adjacence. Les modèles dits cellulaires acceptent des cellules de formes variées, souvent des polygones simples en dimension 2 et des polyèdres en dimension 3. Plus spécialisés, les modèles simpliciaux restreignent les cellules représentées aux simplexes (triangles, tétraèdres, etc.) pour faciliter ou accélérer certains traitements.

Les autres modèles utilisés sont essentiellement des représentations implicites [128, 118, 106]. Les arbres CSG combinent des primitives géométriques et des opérations ensemblistes, pour définir des objets complexes sous forme d'arbres de constructions. Les modèles utilisant un partitionnement spatial [115], comme les octrees ou les arbres BSP, définissent de manière hiérarchique les points de l'espace appartenant à l'objet modélisé. Ces deux types de représentation se révèlent très efficaces pour certains algorithmes, comme la localisation de points ou la détection de collisions. Par contre, ils supportent une classe réduite de traitements du fait de leur incapacité à modéliser certaines entités. Par exemple, les sommets, arêtes ou faces des objets décrits par un arbre CSG ou BSP, ainsi que leurs relations de voisinage, ne sont pas directement accessibles et doivent être calculés en mettant en œuvre des algorithmes complexes [4, 70]. Ainsi l'usage

d'arbres CSG se limite aujourd'hui principalement à la gestion d'historiques de construction en CAO et celui des hiérarchies spatiales à l'accélération d'algorithmes spécifiques.

1.1 Représentation par les bords et modèles ordonnés

Dans le cadre de la représentation par les bords, les relations d'adjacence et d'incidence entre cellules, éventuellement de dimensions différentes, définissent la structure ou la topologie de la subdivision d'un objet géométrique. Beaucoup de modèles, et notamment les modèles simpliciaux, encodent cette structure topologique dans un graphe d'incidence. Parmi ceux-ci, citons le modèle des arêtes radiales de Weiler [135, 134] et ses variations [65, 37, 63, 138], le modèle des Selective Geometric Complexes de Rossignac [120], et concernant les modèles simpliciaux, les modèles proposés par Paoluzzi [111], Kobelt et Seidel [21] ou plus récemment par De Floriani et al. [41, 40, 42].

Simples et permettant de définir un large domaine de représentation, englobant les modèles dits *non variétés*, ces approches souffrent par contre de deux limitations importantes. D'une part, les requêtes topologiques y sont peu efficaces, parce que seule une partie des relations peut être stockée et que certains parcours nécessitent des recherches dans des listes non ordonnées. Pour améliorer l'efficacité de ces modèles, certains ont proposé de stocker de manière redondante les relations d'adjacence fortement utilisées. Efficace cette approche a cependant deux conséquences gênantes. Elle augmente fortement leur coût mémoire et rend l'implantation d'opérateurs topologiques beaucoup plus complexes et la vérification des contraintes d'intégrité très difficile.

Pour pallier à ces deux limitations, les modèles dits ordonnés ont été introduits déjà dans les années 60 et leur usage généralisé dans les années 80 et 90. Leur domaine de représentation est volontairement limité aux subdivisions de pseudo-variétés, orientables ou non, à bord ou fermées, couramment utilisées en modélisation géométrique. Dans ce cadre, les relations topologiques sont fortement contraintes et, en conséquence, leur combinatoire clairement définie. Les modèles ordonnés mettent à profit ces contraintes et l'existence d'orientations, même locales, au sein des cellules pour proposer un encodage de la topologie qui les rend optimaux en termes de coût mémoire, de parcours et d'accès aux relations de voisinages.

Ces notions ont d'abord été exploitées en mathématique, dans le domaine de la topologie combinatoire par Edmonds [52], Jacques [74], Vince [131], Tutte [130] ou Brian [19]. Ils ont plus tard été repris dans le cadre de la modélisation géométrique, généralisés en dimension n et rendus attractifs avec les travaux de Dufourd [49, 50] pour les cartes, et de Lienhardt [96, 97] avec le modèle des G-cartes, pour les variétés non orientables. Une réflexion similaire a été menée par Brisson dans [20] qui introduit le modèles des *cell-tuples*. Des travaux similaires ont été menés à la même époque par Arquès [3] et Spehner [125] introduisant les pavages, représentation duale des 3-cartes. D'autres modèles équivalents aux cartes ou G-cartes existent comme ceux des arêtes ailées, des *half-edges* ou des *quad-edges*. Les cartes ont l'avantage, par rapport à tous ces modèles, de fournir un cadre uniforme, défini de manière homogène en toute dimension.

Les modèles des cartes combinatoires, des cartes généralisées ou des hyper-cartes [34, 35, 36] ont en commun d'être définis de manière algébrique, indépendamment de toute implantation. Ainsi une carte, une G-carte ou une H-carte est un ensemble de brins munis de permutations, notées habituellement α_i, et de contraintes d'intégrité. Les brins sont souvent interprétés géométriquement comme des demi-arêtes orientées, cousues entre elles pour former des faces, des volumes, etc., dont l'assemblage forme une partition des objets à modéliser. En dimension n, il est commode de voir un brin comme un n-uplet représentant les cellules auxquelles il appartient (un sommet, une arête, une face, etc.). Les permutations α_i modélisent les relations d'adjacence entre cellules. Leurs contraintes d'intégrité permettent de contrôler le domaine de validité et la cohérence topologique des objets construits.

Ici, contrairement aux modèles basés sur des graphes d'incidence où les cellules sont explicitement encodées, une cellule est simplement définie par l'ensemble des brins qui la représente. Ainsi,

accéder aux brins d'une cellule permet de parcourir les cellules de son bord ou les cellules qui lui sont adjacentes, par exemple en dimension 2 le parcours d'une face donne simultanément accès à ses arêtes, aux sommets incidents et aux faces adjacentes. Cette approche rend ces modèles très performants tant du point de vue du coût mémoire, grâce à l'absence de redondance, que du point de vue algorithmique. Les structures combinatoires et notamment les modèles topologiques qui dérivent des cartes sont au cœur du projet de recherche présenté dans ce mémoire.

Ma thèse proposait une approche innovante pour décrire et prouver des algorithmes en géométrie algorithmique exploitant pleinement le formalisme de ces structures [29]. Après avoir spécifié algébriquement différents opérateurs topologiques et géométriques pour les cartes de dimension 2 et 3, j'avais proposé un système de réécriture pour formaliser leur raffinement, c'est-à-dire la subdivision de leurs cellules jusqu'à disparition de toutes intersections ou recouvrements. Ce raffinement était couplé à une reconstruction des relations d'adjacence, pour obtenir des objets dont la topologie était garantie. Ces travaux ont été conduits d'abord en dimension 2 [23, 24], où des preuves formelles ont été obtenues [26] en étudiant la convergence des systèmes de réécriture, puis en dimension 3 [25, 27]. Nous avions ainsi obtenu, par dérivation de programmes, des algorithmes robustes pour calculer l'union ou l'intersection d'objets formés de cellules polyédriques.

Après ces débuts, mêlant spécifications formelles et algorithmique géométrique, mes travaux se sont diversifiés et élargis à d'autres aspects de la modélisation et de l'algorithmique géométrique. Au cœur de ceux-ci se trouvent toujours les cartes combinatoires. J'ai cherché, à travers différents thèmes, à en proposer des extensions visant à élargir leur domaine de représentation et leurs usages. J'ai également essayé de mettre à profit leurs propriétés combinatoires en abordant sous des angles nouveaux divers problèmes d'algorithmique géométriques. Cela m'a conduit à explorer quatre principaux axes de recherche décrits plus en détail ci-après.

1.2 Complexes cellulaires et modèles non variétés

Les cartes ou G-cartes permettent de modéliser les quasi-variétés de dimension n de manière très efficace. Cependant pour certaines applications en CAO ou CFAO, la possibilité de représenter des objets qui ne sont localement pas des variétés reste essentielle [31, 105, 103, 93]. Ces modèles dits *non variétés* offrent un domaine de représentation plus large, autorisent des sommets singuliers et conduisent à une représentation simplifiée des modèles cellulaires qui permet de combiner ensemble des objets aux dimensions variées, comme des courbes, des surfaces ou des solides.

Dans ce cadre, différents buts sont poursuivis : la définition de nouveaux modèles ou schémas de représentation [63, 122, 112], la spécification d'opérateurs pour des modèles particuliers [64, 134, 104] ou l'implantation de fonctionnalités de haut niveau comme la modélisation par *offset* [93] ou par balayage [136] et les opérations booléennes non régularisées [37, 66].

Les premiers travaux significatifs sur la modélisation de complexes cellulaires sont ceux de Weiler [135] avec la structure des arêtes radiales qui reste un standard utilisé dans de nombreuses applications de CAO. Ce modèle mélange la représentation explicite des cellules de différentes dimensions, avec des entités topologiques qui leur sont associées et qui portent leurs relations d'adjacence (usuellement dénommées *face-use*, *loop-use*, *edge-use* et *vertex-use*).

Avec cette approche, toutes les cellules et relations sont explicitement encodées. Les structures de données en dérivant sont lourdes et très complexes, notamment parce qu'elles mélangent la structure topologique et les plongements géométriques. En terme de génie logiciel, leur gestion reste difficile et implique le développement de nombreux opérateurs devant être déclinés pour chaque type de cellules et d'adjacence. D'autre part, les recherches et parcours dans le voisinage de cellules sont loin d'être optimaux, à cause de l'absence d'ordre dans certaines listes d'adjacence.

Des améliorations ont été proposées par Gursoz et al. [65] avec la *Tri-cyclic Cusp Structure* se focalisant sur les sommets. Yamaguchi et Kimura [138] ont proposé une structure similaire, les *Coupling Entities*, dans le but d'accélérer les parcours, au dépend des coûts mémoire. Plus tard,

une version plus compacte, la *Partial Entity Structure*, a été proposée dans [94]. Les gains en termes de coûts mémoire se font ici au détriment de la création de nouvelles entités topologiques qui conduisent à un graphe d'adjacence encore plus complexe.

Peu d'autres approches ont été proposées pour la modélisation des objets non variétés. Les travaux de Rossignac [120] sur les SGC déjà mentionnés ou de Lienhardt sur les chaînes de cartes ou les complexes cellulaires [53] ont été peu exploités. Ces modèles possèdent une topologie précisément contrôlée et conduisent à des structures à base de graphe d'incidence très complètes, mais malheureusement trop verbeuses pour être exploitées sur des données complexes.

Les travaux sur les structures simpliciales ont été plus fructueux. Lienhardt a étudié des modèles basés sur des ensembles semi-simpliciaux [91] très généralistes. La plupart des autres travaux portent cependant sur des structures de données plus concrètes, modélisant des complexes triangulaires ou tétraédriques, comme ceux de Rossignac [119] ou de De Floriani [41, 40]. Dans ces modèles seules les cellules de dimension maximale (triangles ou tétraèdres) et les sommets singuliers sont représentés. Cela conduit à des structures de données en général très compactes et relativement efficaces, les parcours pouvant être optimisés en fonction des applications visées par l'ajout de relation spécifiques. Ces modèles restent cependant des graphes d'adjacences, avec les défauts déjà mentionnés et notamment l'absence de certaines cellules sur lesquelles aucune information ne peut donc être stockée.

En pratique, les complexes cellulaires usuellement manipulés peuvent être décomposés en portions de courbes, surfaces ou volumes, reliées entre elles autour de sommets singuliers. En décomposant habilement ces objets, il est possible de les modéliser par un ensemble de variétés auxquelles sont ajoutées des relations de collage spécifiques. Pesco et al ont proposé une méthode de décomposition utilisant ce principe, limitée au complexes cellulaires de dimension 2 [112].

Généralisant cette approche, le modèle des X-maps, pour cartes étendues, ajoute une nouvelle relation, nommée θ, permettant de lier les sommets d'une carte autour de points singuliers. Les composantes connexes de la carte privée de la relation θ forment une décomposition des objets en variétés qui n'est bien sur pas unique. Nous avons proposé, dans [28], un mécanisme utilisant une représentation duale des cartes et permettant de représenter les composantes connexes de dimensions différentes en ne stockant pas les points fixes des relations α_i. L'ensemble fournit un modèle très compact, jusqu'à 4 fois moins coûteux en mémoire que celui des arêtes radiales, bénéficiant de toute l'efficacité des cartes sur les portions correspondant à des variétés. Les détails peuvent être vus dans l'article présenté dans l'annexe C.

1.3 Maillages multirésolution

La représentation multirésolution d'objets géométriques suscite un intérêt croissant dans de nombreux domaines de l'informatique graphique, et notamment en modélisation géométrique. Dans le cadre de la représentation par les bords, une surface est décrite par un maillage, c'est-à-dire une subdivision finie composée de sommets, d'arêtes et de faces. Dans une représentation multirésolution, cette surface n'est plus définie par un maillage unique, mais par une série de maillages imbriqués, et par les règles permettant de passer de l'un à l'autre. Ces maillages constituent les différents niveaux de résolution de l'objet représenté. La représentation est dite adaptative si la résolution maximale n'est pas la même pour toutes les zones de l'objet.

Les modèles multirésolution trouvent de nombreuses applications allant de l'analyse multirésolution pour le filtrage ou la compression de maillages [102], à leur transmission progressive [126], en passant par l'affichage selon le point de vue ou l'édition multirésolution [141, 83]. Les surfaces de subdivision multirésolution [139] sont une des méthodes permettant de construire des surfaces multirésolution en partant d'un maillage grossier qui est raffiné jusqu'à l'obtention d'une surface suffisamment lisse. De nombreux outils ont été proposés pour manipuler ces surfaces, comme l'insertion d'arêtes vives [12], l'application d'opérations booléennes [11] ou des opérateurs de collage d'éléments de surface [13].

En général, les auteurs de ces outils se focalisent sur le traitement de la géométrie et moins sur les structures de données sous-jacentes. Les modèles utilisés ne sont alors malheureusement pas les plus optimaux. La structure de données la plus souvent utilisée est une forêt de *quadtrees* dérivant naturellement de la hiérarchie de faces générée par les algorithmes de subdivision [43]. Les structures à base de quadtrees sont limitées aux maillages triangulaires ou carrés. Les requêtes d'adjacence y sont exécutées en $O(log(n))$ et même si ce coût est borné par le nombre de niveau de résolution, il reste gênant pour les traitements géométriques usuels. Des solutions ont été proposées [121, 92], mais elles impliquent d'autres limitations comme l'impossibilité d'une représentation adaptative.

A contrario, de nombreux maillages sont aujourd'hui issus de processus d'acquisition 3D. Ceux-ci sont généralement très denses et leur connectivité ne présente pas de régularité particulière. Dans ce cadre, des travaux ont été menés afin de construire une hiérarchie naturelle permettant de leur appliquer les algorithmes de traitement multirésolution mentionnés plus haut. Ces modèles multirésolution sont générés à partir d'une représentation détaillée qui est simplifiée peu à peu en utilisant des opérateurs de contraction. Cette approche a été utilisée pour les multi triangulations proposées par Puppo et al. [45] et pour les maillages progressifs proposés par Hoppe [71, 114].

En pratique ces représentations ne font pas appel à des structures multirésolution spécifiques. Elles travaillent avec le maillage fin et une liste d'opérations de simplifications qui sont évaluées à la demande. Ces opérations peuvent être organisées dans un arbre de dépendances pour être appliquées de manière adaptative. Des méthodes accélératrices ont été proposées pour passer d'un niveau de détail à l'autre [72, 44]. Elles utilisent un maillage courant et maintiennent un *front* marquant, dans l'arbre des opérations, celles qui sont appliquées à un instant donné. Elles permettent alors une navigation adaptative dans les différents niveaux de résolution. Cependant l'accès aléatoire aux différentes résolutions, nécessaire pour l'analyse ou l'édition multirésolution, reste très coûteux.

Avec la thèse de Pierre Kraemer, nous avons proposé un nouveau modèle, les cartes multirésolution ou MR-maps [86, 84, 87], basé sur les cartes combinatoires et en héritant la formalisation, la généricité et l'efficacité. Le modèle des MR-maps est adaptatif c'est-à-dire qu'il permet un raffinement à une profondeur variable suivant les zones de l'objet modélisé. Dans les structures à base d'arbres généralement utilisées dans ce contexte, l'adaptabilité engendre des trous topologiques qui sont évités avec les MR-maps. Nous avons montré que le coût mémoire des MR-maps était similaire à celui des quadtrees dans le cas général et adaptatif. Alors que l'efficacité des parcours des différents niveaux de résolution est grandement améliorée.

Nous avons proposé le plongement des MR-maps sur des surfaces de subdivision. Ce nouveau type de plongement, jamais proposé pour des cartes combinatoires, possède dans ce contexte de nombreux avantages. Les MR-maps supportent un large éventail de schémas de subdivision : les schémas primaux (découpant les faces) ou duaux (éclatement de sommets), et même des schémas plus originaux et non encore utilisés en multirésolution comme le schéma $\sqrt{3}$ ou le schéma quad/triangle mélangeant facettes triangulaires et rectangulaires [85].

Les cartes multirésolution ont également été utilisées pour implanter de manière générique des maillages progressifs. Elles permettent de construire une hiérarchie de maillages réellement multirésolution, c'est-à-dire dont tous les niveaux sont accessibles, à partir d'une succession de contractions de cellules.

1.4 Détection de collisions

Une carte combinatoire décrit une subdivision en cellules d'une variété donnée. Lorsque celle-ci est plongée sans intersection ni recouvrement, nous dirons *bien plongée*, elle réalise une partition de son espace de plongement. Par exemple, une 2-carte bien plongée dans $\mathbb{R}^2$ décrit une partition du plan et une 3-carte plongée dans $\mathbb{R}^3$, une partition de l'espace. Cette notion permet de définir

une décomposition naturelle d'un objet plongé dans $\mathbb{R}^3$. Cette décomposition spatiale épouse implicitement les frontières et structures internes d'un objet correspondant aux cellules de la 3-carte le modélisant. Cette idée simple nous a permis d'aborder de manière nouvelle un problème bien connu en géométrie algorithmique : la détection de collisions [75, 99, 127].

De manière générale, la complexité de la recherche de collisions entre les primitives d'une scène dépend de manière quadratique du nombre de primitives présentes. Pour accélérer ces traitements, il faut limiter le nombre de couples de primitives à tester. La méthode la plus répandue pour cela consiste à utiliser une décomposition spatiale, souvent hiérarchique, de la scène. Des hiérarchies de boites englobantes de natures diverses ont été utilisées, comme les AABB (*Axis-aligned bounding box*) [33], les OBB (*Oriented Bounding Box*) [59] ou les k-dop (k *Discrete Oriented Polytope*) [82].

Durant une phase dite *grossière*, les couples de primitives appartenant à une même boite englobante sont recherchés. Lorsqu'un couple de primitives potentiellement en collision est trouvé, la phase dite *exacte* commence. Différentes méthodes permettent de savoir si les primitives concernées sont complètement séparées ou en collision. Ces méthodes permettent d'obtenir, de manière plus ou moins détaillée, les informations de contact entre primitives ou la géométrie de leurs interpénétrations éventuelles.

Lorsque les primitives sont des polyèdres convexes, l'approche classique consiste à rechercher les couples de points les plus proches. Les méthodes GJK [58] et Lin-Canny [98], font références dans le domaine. Dans le cas général, une décomposition en polyèdres convexes est généralement utilisée. Des approches stochastiques [67] ou évolutionnistes [76] ont également été proposées. Au lieu de considérer l'ensemble des couples de sommets entre primitives, elles échantillonnent ceux-ci en sélectionnant un nombre réduit de couples de sommets représentatifs qui évoluent au cours du temps. Ces méthodes ne sont, par nature, plus réellement exactes. Elles sont efficaces pour des objets complexes, concaves, mais dont le nombre est réduit, le nombre d'échantillons dépendant du nombre d'objets de manière quadratique.

Les méthodes dites multirésolution se situent au carrefour des phases grossières et exactes. Dans [110], un système appelé CLODs, pour *Contact Level Of Details*, est décrit. Les recherches de collisions sont effectuées à faibles résolutions lorsque les zones de contactes sont importantes et sont étendues aux résolutions détaillées en cas de contacts fins. Ces idées se retrouvent également dans les travaux de Hubbard [73], introduisant la notion de temps critique. Ici la résolution utilisée pour détecter une collision dépend du temps alloué à ce calcul.

Quelques travaux récents mettent à profit la puissance de calcul des processeurs graphiques pour effectuer un grand nombre de tests géométriques sur GPU [60, 62, 54]. Mais dans le cas de scènes complexes, l'utilisation d'une hiérarchie de volumes englobants reste nécessaire. De plus la précision des résultats dépend de la résolution graphique utilisée et est donc inversement proportionnel à la complexité de la scène.

Même si la détection de collisions est un domaine qui a été fortement étudié, les méthodes existantes atteignent leurs limites lorsque les objets considérés sont déformables ou lorsque leur topologie est amenée à changer lors de découpes ou de collages, par exemple. Dans ces cas, les boites englobantes et la hiérarchie volumique qui les contient doivent être continuellement mises à jour durant la simulation. Cela réduit d'autant les bénéfices sur les temps de calcul. Ces surcoûts réduisent de manière drastique la complexité des simulations réalisables en temps réel.

C'est notamment le cas, dans le cadre d'applications médicales, par exemple pour les simulations chirurgicales où des outils (scalpels, cathéters, fils de suture, caméras, etc.) sont en mouvement au sein d'un environnement complexe fortement déformable, formé par le corps humain et ses organes. Dans le cas de laparoscopies, les difficultés sont encore plus importantes car certains des objets en mouvement sont complètement inclus dans les autres ce qui est relativement mal supporté par les hiérarchies de boites englobantes classiques.

Le but premier des structures hiérarchiques présentées ci-dessus est de partitionner l'espace en cellules, ou boites réduites, dans lesquelles un nombre restreint de primitives se trouvent

simultanément. Certains travaux ont tenté de remplacer ces hiérarchies par une partition fixe de l'environnement qui se déforme avec la simulation. La trajectoire des objets en mouvement est suivie au sein de ces cellules, afin de savoir à tout moment quelles sont les primitives situées dans une cellule donnée. Les travaux de Geiger [56] et Lenoir [95] utilisent ce principe pour la simulation d'endoscopies ou d'angioscopies. Ces deux travaux se limitent cependant à la modélisation de réseaux vasculaires et utilisent une décomposition en tétraèdres assez lourde, pour le premier, et une décomposition en cellules hexaédriques organisées dans un arbre fixe, pour le second. Ils considèrent uniquement le déplacement de cathéters modélisés par des lignes brisées et, de plus, ne supportent pas les changements topologiques dans la scène.

Avec la thèse de Thomas Jund, nous avons proposé un nouveau système de détection de collisions, permettant de suivre les déplacements de maillages quelconques au sein de scènes complexes. L'environnement dans lequel les mobiles sont en déplacement est décomposé en polyèdres convexes et modélisé par une 3-carte. En considérant que les déformations géométriques engendrent peu de changements dans la topologie de cette subdivision, nous obtenons une partition volumique de la scène qui s'adapte naturellement à ses propres déformations lors de la simulation.

Les mobiles en déplacement sont modélisés par des maillages qui peuvent être quelconques et sont usuellement surfaciques. La première brique de notre système de détection consiste à suivre le déplacement des sommets de ces maillages, en mettant à profit les relations d'incidence et d'adjacence présentes dans les 3-cartes. Nous avons proposé un mécanisme de prédiction permettant d'exploiter la cohésion temporelle de la simulation. Son efficacité est assurée par la structure combinatoire des 3-cartes et leur décomposition implicite en tétraèdres orientés optimisant le nombre de tests géométriques à effectués [78, 77]. Ce système supporte naturellement les déformations de l'environnement et des mobiles, et permet une gestion très efficace des changements topologiques grâce à des opérateurs locaux impliquant un nombre contrôlé de mises à jour.

Ces résultats ont été étendus au suivi des arêtes des maillages en déplacement [80, 79]. Leurs mouvements sont décomposés en déplacements élémentaires formant des triangles qui balayent l'espace environnant. En utilisant des arguments géométriques sur la convexité de ces déplacements et des cellules parcourues, il est possible d'obtenir une information de collision ou de contact précise, avec peu de calculs supplémentaires. Nous avons ainsi pu expérimenter notre système avec deux types de simulations temps réel très contraignantes : l'insertion d'un cathéter dans un réseaux vasculaire déformable et le déplacement de solides déformables dans un environnement lui-même déformable et dont la topologie change.

1.5 Génération de maillages à partir d'images discrètes

A l'opposé et en amont de la simulation, les applications médicales utilisent des données obtenues par imagerie scanner ou IRM permettant d'acquérir, de plus en plus facilement, des images 3D précises des structures anatomiques d'un patient. Or, pour les applications visées, ces images brutes ne suffisent plus. Elles nécessitent une représentation du corps humain plus élaborée. En général un modèle géométrique maillé est nécessaire. Ce modèle doit être constitué d'un ensemble de surfaces, pour le rendu ou l'interaction, alors qu'un maillage volumique est nécessaire pour les applications nécessitant une simulation physique.

Les modèles géométriques générés doivent non seulement fournir une approximation géométrique précise des frontières des objets discrets, mais également définir une topologie cohérente, pour que les interactions entre structures anatomiques puissent être correctement rendues ou simulées. En termes de modèle géométrique, cela signifie que chaque structure anatomique doit posséder un maillage définissant une surface fermée et orientée, sans singularité ni recouvrement, qui la sépare de son complémentaire dans l'image [124]. De plus, les structures anatomiques présentes dans l'image étant adjacentes ou imbriquées, leurs différents maillages doivent être compatibles au niveau des surfaces de contact et des jonctions.

Reconstruction surfacique : Il y a peu de méthode de reconstruction de maillages qui gèrent réellement des images contenant plusieurs objets imbriqués. Les plus populaires sont les extensions de l'algorithme du *Marching-Cube*. Une première généralisation a été proposée dans [68] et une seconde dans [137]. Elles ont étendu la table de correspondance utilisée dans le Marching-Cube pour supporter d'abord trois, puis jusqu'à 8, objets incidents à une même cellule.

Le problème, avec les méthodes issues du Marching-Cube, est que leur résultat contient un nombre très important de triangles qui présentent souvent des artéfacts prenant la forme de marches d'escalier. Les maillages obtenus pour des images médicales standard peuvent compter jusqu'à plusieurs millions de triangles. Cela oblige à effectuer une seconde passe de traitement durant laquelle les maillages sont ré-échantillonés, c'est-à-dire simplifiés et lissés, en utilisant différents types de filtres. Ces optimisations sont coûteuses en temps de calcul et restent très sensibles au bruit présent dans les images voxels de départ.

Les méthodes de type *Dual Contouring* sont moins répandues, mais plus simples. Elles travaillent directement sur des images multi labels. Pour cela, elles opèrent sur la grille duale obtenue en décalant l'image d'un demi voxel dans chaque direction. Ces méthodes ont d'abord été proposées pour des images binaires [57, 109]. Elles fournissent des maillages quadrilatères dont les faces sont les rectangles correspondant aux frontières inter-voxels. Comme pour le Marching-Cube, ces méthodes nécessitent un ré-échantillonage ultérieur.

Des travaux récents [8, 116, 14] ont adaptés ces méthodes duales au traitement d'images médicales. Les maillages obtenus ne sont pas des variétés, notamment lorsque des voxels de même label sont 18- ou 26-connexes. Pour éviter cela ces travaux proposent différentes stratégies de subdivisions des voxels visant à supprimer les cas ambigus. L'idée, à chaque fois, est de trouver un niveau de détail où les surfaces d'objets adjacents sont séparées. Cela permet de générer localement des maillages qui sont bien des 2-variétés. Mais aucune de ces méthodes ne garantit l'absence de singularités ou la cohérence topologique de manière globale. De plus elles ne gèrent pas explicitement les jonctions durant la phase de décimation ce qui peut entraîner l'apparition de trous ou de superpositions.

Reconstruction volumique : Pour la génération de maillages volumiques trois types d'approches sont utilisées. Les premières et plus anciennes, dites de *packing*, partent d'un maillage de la surface et font progresser, à partir de là, un front de tétraèdres qui remplit peu à peu l'intérieur des volumes [32]. Relativement simples en dimension 2, ces méthodes se révèlent beaucoup plus complexes en dimension 3. En particulier, les contraintes sur les tétraèdres centraux sont si fortes quelles posent des problèmes quant à la convergence et à la qualité des maillages obtenus.

Les secondes sont basées sur un partitionnement régulier de l'espace, en général hexaédrique, dont les cellules sont subdivisées en tétraèdres après sélection [90]. Les tétraèdres coupant le bord des objets à mailler sont subdivisés au besoin. Ces méthodes ont l'avantage d'être très rapides, mais produisent des maillages de faible qualité, et ont été abandonnées au profit de méthodes apportant une garantie théorique sur la qualité des maillages.

Le troisième type d'approche, dite de *Delaunay*, est maintenant le plus répandu. Ces méthodes sont liées au diagramme de Voronoï et procèdent de manière itérative [123, 108]. Partant d'un maillage grossier, des sommets internes sont ajoutés tant que certains critères géométriques ne sont pas satisfaits. A chaque étape, le critère de Delaunay est vérifié et le maillage corrigé localement.

Malheureusement, en dimension 3, ce critère de Delaunay ne suffit pas à empêcher l'apparition de tétraèdres dégénérés, nommés *slivers*. De plus, la construction et le choix des sommets internes aux zones à mailler ne sont pas simples et il y a peu de moyens d'assurer que les zones de contacts ou jonctions soient correctement discrétisées. Pour palier à ces limitations, ces approches ont été enrichies par des étapes d'optimisation du maillage construit, comme par exemple dans [129]. Cela donne de très bons résultats pour des maillages issus du monde de la CAO, mais la prise en compte des 1-jonctions et des imbrications de plusieurs volumes pose encore problème. Enfin, les

temps de calculs de ces dernières méthodes sont beaucoup plus longs, les étapes d'optimisation pouvant être très coûteuse.

Des méthodes de maillage issues de la géométrie discrètes avaient été étudiées dans notre équipe par Dobrina Boltcheva durant sa thèse encadrée par Dominique Bechmann et soutenue en 2007. Elle avait développé un algorithme de reconstruction surfacique basé sur la notion de graphe de Voronoï discret. Elle y définissait un *micro-modèle* représentant une subdivision de surface dans une image voxel dont chaque arête était de longueur 1. Ce modèle théorique, trop coûteux pour être implanté, permettait de décrire un algorithme de construction du diagramme de Voronoï discret et de son dual le Delaunay discret utilisé pour la triangulation du résultat.

Avec les travaux menés par Cyril Kern [17], nous avons proposé un codage permettant de représenter ce micro modèle par des cartes combinatoires de dimension 3 définissant ainsi des variétés volumiques sur les images voxel. Les micro-cartes sont voisines des pyramides de cartes introduites pour la segmentation d'images. Elles sont définies de manière implicite en utilisant un octet par voxel ce qui les rend exploitable en pratique sur des données réelles. Cette nouvelle représentation nous a permis d'étendre les résultats obtenus par Dobrina Boltcheva aux images multi labels, c'est-à-dire contenant plusieurs objets adjacents, tout en gérant correctement les zones de contact. Elle a aussi permis d'accélérer grandement les calculs ce qui les rend maintenant compétitifs face aux algorithmes cités précédemment comme celui du Marching-Cube.

Avec ce modèle, nous avons également proposé des outils pour définir la notion de graphe de Voronoï 3D discret. Nous avons ainsi obtenu un premier algorithme de reconstruction et de génération de maillages volumiques à partir d'une image voxel, entièrement discret et garantissant la cohérence topologique du résultat.

1.6 Plan du mémoire

Les trois principaux axes de recherche présentés ci-dessous sont développés dans la suite de ce mémoire. Le premier chapitre décrit les cartes multirésolution et leur usage pour l'implantation de surfaces de subdivision et de maillages progressifs. Une étude précise des coûts mémoire et une comparaison avec les modèles classiques est proposée. Dans le second chapitre, le système de détection de collisions basé sur l'exploitation de cartes volumiques est détaillé. Nous présentons une étude algorithmique du suivi de particules et les différentes réponses qui peuvent être utilisées dans une simulation physique. Le troisième chapitre, présente les micro-cartes et leur utilisation pour le maillage d'images discrètes. Les travaux sur les cartes étendues et les maillages non variétés ne sont pas détaillés, ainsi que ceux portant sur le maillage et la reconstruction des réseaux vasculaires. Les articles correspondants sont cependant fournis en annexe, pour information.

Enfin, le quatrième et dernier chapitre propose une synthèse des perspectives de recherche liées à l'ensemble de ces travaux. J'y expose un programme de recherche à plus ou moins longs termes, mélant algorithmique géométrique et structures combinatoires pour les représentations multi-échelles, dans le cadre de la modélisation géométrique, la simulation physique et le traitement de la géométrie.

Chapitre 2

Cartes multirésolution

Les travaux présentés dans ce chapitre ont été réalisés durant la thèse de Pierre Kraemer [88] encadrée par Dominique Bechmann et moi-même. Ils ont donné lieu à une publication en revue internationale [87], à une publication en revue nationale [84] et à deux communications lors de conférences internationales [85, 86].

Il s'agissait de définir un nouveau cadre pour la représentation de maillages multirésolution, mettant à profit les qualités déjà mentionnées des cartes combinatoires. Les cartes multirésolution sont définies section 2.1. Ce modèle définit une hiérarchie de cartes imbriquées les unes dans les autres et représentant les différents niveaux de résolution d'un objet modélisé. Deux applications ont été étudiées. La première, présentée section 2.2, concerne l'association des cartes multirésolution et des surfaces de subdivisions. Elle permet une représentation uniforme d'un grand nombre de shémas de subdivision proposés dans la littérature, leur extension à des maillages quelconques et leur mélange au sein d'un même mailage. La seconde application, présentée section 2.3, concerne l'usage des cartes multirésolution pour l'encodage des maillages progressifs. Enfin, section 2.4, nous discutons de l'implantation des cartes multirésolution et les comparons à différents modèles classiques en termes de coûts mémoire et d'efficacité.

2.1 Cartes combinatoires multirésolution

Nous commençons par quelques rappels sur les cartes combinatoires de dimension 2, afin de fixer les notations et conventions utilisées dans le reste du chapitre. Puis donnons une définition

générale des cartes multirésolution dans des versions primales et duales.

2.1.1 Les cartes combinatoires primales et duales

Une 2-carte est un triplet (B, α_0, α_1) où B est un ensemble de brin équipé de deux permutations, avec la contrainte que α_0 soit une involution sans point fixe, c'est à dire une permutation telle que $\forall x \in B, \alpha_0(\alpha_0(x)) = x$ et $\alpha_0(x) \neq x$. Ce modèle permet de représenter la topologie des subdivisions de surfaces orientables fermées, dont les composantes connexes modélisent des solides usuels. Les brins sont classiquement interprétés comme des demi arêtes. Dans ce cas, deux brins en relation par α_0 forme une arête, et la permutation α_1 lie les brins autour de sommets. La figure 2.1 présente les conventions graphiques adoptées pour la représentation des brins et des relations.

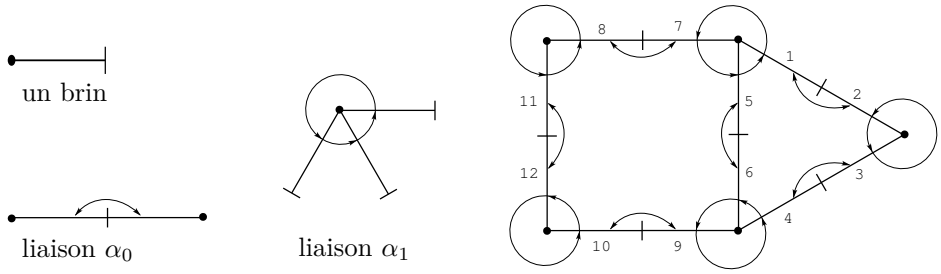


FIGURE 2.1 – Conventions graphiques pour les cartes primales et une 2-carte composée de cinq sommets : {1, 7, 5}, {2, 3}, {4, 6, 9}, {8, 11}, {10, 12} ; six arêtes : {1, 2}, {3, 4}, {5, 6}, {7, 8}, {9, 10}, {11, 12} ; et trois faces : {1, 3, 6}, {2, 7, 11, 10, 4}, {5, 9, 12, 8}.

Les cellules de la subdivision sont définies par des ensembles de brins correspondant aux orbites des permutations. Rappelons que pour une permutation σ sur B, l'orbite de $x \in B$ est le sous-ensemble noté $\langle\sigma\rangle(x)$ égal à $\{x, \sigma(x), ..., \sigma^k(x)\}$, où k est le plus petit entier tel que $\sigma^{k+1}(x) = x$. Clairement, tous les éléments de $\langle\sigma\rangle(x)$ ont la même orbite. Une orbite représente l'ensemble des brins atteignables par les applications successives de la permutation σ. Les sommets sont définis par l'orbite $\langle\alpha_1\rangle$, les arêtes par l'orbite $\langle\alpha_0\rangle$, et les faces par l'orbite $\langle\alpha_0 \circ \alpha_1\rangle$ (voir figure 2.1).

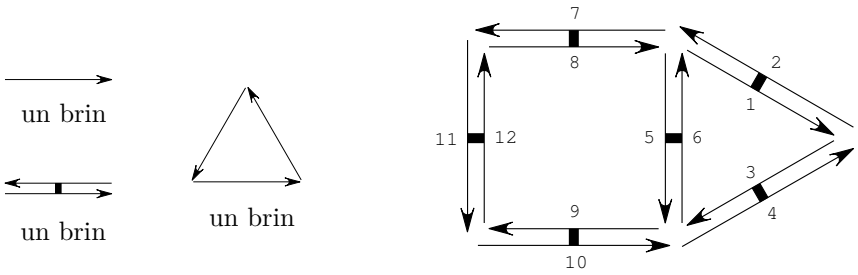


FIGURE 2.2 – Conventions graphiques pour les cartes duales

Il est possible de définir les cartes de manière duale. Cela est interéssant dans certaines applications où les faces jouent un rôle plus important que les sommets, comme dans le cas de certains schéma de subdivisions présentés plus loin. Etant donnée une 2-carte $M = (B, \alpha_0, \alpha_1)$, que nous dirons primale, le triplet $M' = (B, \phi_1, \phi_2)$, avec $\phi_1 = \alpha_0 \circ \alpha_1$, et $\phi_2 = \alpha_0$, est aussi une 2-carte appelée carte duale de M. La relation ϕ_1 est une permutation des brins autour de faces orientées, et ϕ_2 est une involution sans point fixe permettant de coudre ces faces (voir figure 2.2).

Ici, les sommets sont définis par l'orbite $\langle \phi_1 \circ \phi_2 \rangle$, les arêtes par l'orbite $\langle \phi_2 \rangle$, et les faces par l'orbite $\langle \phi_1 \rangle$.

2.1.2 Les cartes multirésolution primales et duales

Les cartes multirésolution sont conçues pour modéliser la topologie d'objets multirésolution. Chaque niveau de résolution est représenté par une carte et en hérite les propriétés. Pour que ces différentes cartes forment réellement une hiérarchie et s'imbriquent naturellement les unes dans les autres, les brins qui les constituent sont réutilisés d'un niveau de résolution à l'autre.

Ainsi, une carte multirésolution est composée de brins introduits progressivement à chaque résolution. Notons B^i, l'ensemble des brins de B qui font partie du niveau de résolution i. Dans une carte multirésolution, de résolution maximale k, ces ensembles forment une suite $\{B^i\}_{i\in[0,k]}$ telle que :

$$B^0 \subset B^1 \subset B^2 \subset \ldots \subset B^k = B$$

L'ensemble B^i est le sous-ensemble de B contenant les brins qui ont été introduits à un niveau inférieur ou égal à i. L'ensemble $N^i = B^i \setminus B^{i-1}$ contient les brins qui ont été introduits au niveau i. L'ensemble de brin B correspond aux brins de la carte décrivant le maillage au niveau le plus fin. Entre deux niveaux de résolution, les relations topologiques peuvent changer. Pour cela, nous enrichissons les cartes par un ensemble de permutations α_0^i et α_1^i définies au niveau de résolution i. De cette manière, une carte multirésolution, de résolution maximale k, est un triplet :

$$M = \left(B, \{\alpha_0^i\}_{i\in[0,k]}, \{\alpha_1^i\}_{i\in[0,k]}\right)$$

tel que pour tout $i \in [0, k]$, le triplet $M^i = (B^i, \alpha_0^i, \alpha_1^i)$ soit une carte.

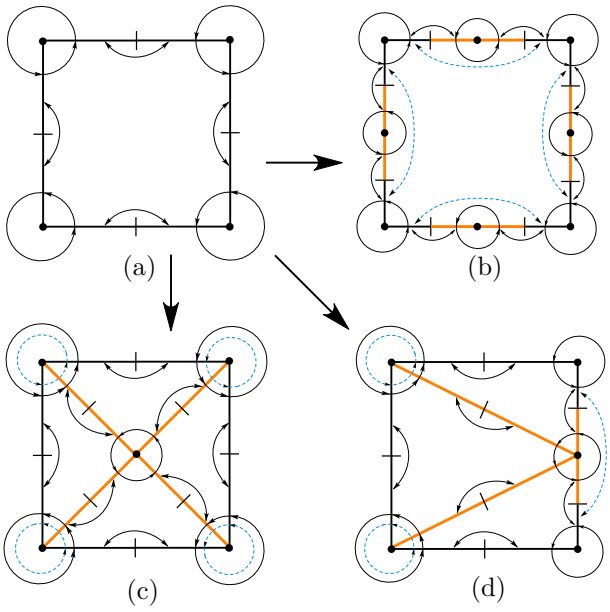


FIGURE 2.3 – Exemples de 2 niveaux successifs de 2-cartes multirésolution

Les cellules d'un niveau de résolution i sont définies dans la carte M^i de la même façon que pour les cartes monorésolution. Ainsi, les sommets du niveau i sont définis par l'orbite $\langle \alpha_1^i \rangle$, les arêtes par l'orbite $\langle \alpha_0^i \rangle$, et les faces par l'orbite $\langle \alpha_0^i \circ \alpha_1^i \rangle$.

La figure 2.3(a) illustre un exemple de 2-carte multirésolution à un niveau i, composée de 4 sommets, 4 arêtes et 2 faces. Les figures 2.3(b), 2.3(c), 2.3(d) illustrent trois exemples de ce

que pourrait être le niveau $i+1$. Dans les trois cas, les brins en orange sont ceux qui ont été introduits entre les deux niveaux, c'est-à-dire ceux appartenant à l'ensemble N^{i+1} ou $B^{i+1} \setminus B^i$. Suivant les insertions de brins effectuées entre les deux niveaux, les relations liant des brins au niveau i ne sont pas nécessairement différentes au niveau $i+1$. Lorsque deux brins liés au niveau i ne le sont plus au niveau $i+1$, un rappel de leur liaison de niveau i est dessiné en pointillés bleus.

Les cartes multirésolution ont également une définition duale. Soit M une carte multirésolution, alors le triplet :

$$M' = (B, \{\phi_1^i\}_{i\in[0,k]}, \{\phi_2^i\}_{i\in[0,k]})$$

avec $\phi_1^i = \alpha_0^i \circ \alpha_1^i$ et $\phi_2^i = \alpha_0^i$, est aussi une 2-carte multirésolution, appelée carte duale de M. Chaque niveau de résolution i, avec $i \in [0,k]$, est représenté par la 2-carte duale $M^i = (B^i, \phi_1^i, \phi_2^i)$. La figure 2.4 illustre en représentation duale, les exemples des figures 2.3(a) et 2.3(d).

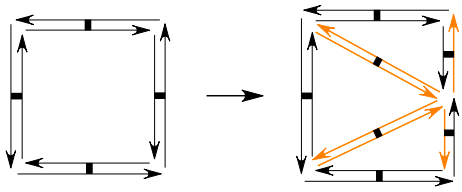


FIGURE 2.4 – 2 niveaux successifs d'une 2-carte multirésolution duale

Dans une carte, le plongement permet d'associer une information géométrique à une cellule topologique. Classiquement, des points sont associés aux sommets d'une 2-carte. Pour les cartes multirésolution, le plongement est lui aussi paramétré par le niveau de résolution. Dans les applications suivantes, chaque sommet de niveau de résolution $i \in [0,k]$ est associé à un point disctinct, ce qui permet à la géométrie des maillages d'évoluer d'un niveau de résolution à l'autre, comme c'est l'usage.

2.2 Application aux surfaces de subdivision

Avant de montrer comment les cartes multirésolution permettent de représenter efficacement des surfaces de subdivision, nous rappelons, dans un premier temps, le principe des surfaces de subdivision ainsi que de leur extension multirésolution.

2.2.1 Les surfaces de subdivisions

Le principe des surfaces de subdivision est de définir une surface comme la limite d'une séquence de maillages raffinés successivement. Un schéma de subdivision définit la manière d'appliquer ce raffinement au maillage (voir figure 2.5). On peut distinguer ici deux étapes : d'abord, la topologie du maillage est raffinée, puis de nouvelles informations géométriques sont calculées à partir du maillage de départ et associées au maillage obtenu. Différents schémas génèrent des surfaces aux propriétés différentes.

Le processus de calcul de la géométrie d'un schéma de subdivision est habituellement classé en deux familles, appelées interpolante et approximante. Dans les deux cas, les informations géométriques associées aux sommets sont calculées localement pour chaque sommet en appliquant un masque sur les positions des sommets voisins dans le maillage de départ. Dans un schéma approximant, les positions de tous les sommets du nouveau maillage sont calculées et se rapprochent de la surface limite. Dans un schéma interpolant, les sommets du maillage de départ se situent déjà sur la surface limite, et seules les positions des nouveaux sommets insérés sont calculées.

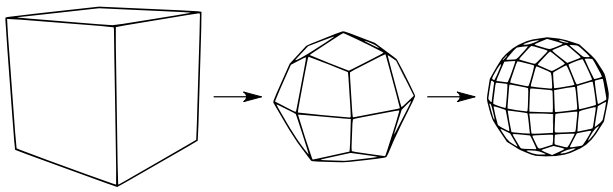


FIGURE 2.5 – 2 itérations de subdivision sur une sphère topologique

Le processus de raffinement de la topologie d'un schéma de subdivision est également classé en deux familles, appelées primale et duale. Dans les shémas primaux les faces des maillages sont subdivisées, alors que leurs sommets ne changent pas. Au contraire, dans les shémas duaux, les faces ne changent pas et les sommets sont éclatés. Nous allons voir dans la suite les différences entre ces schémas, et comment les modèles que nous avons définis dans la section précédente permettent de les supporter.

Le principe de l'extension multirésolution des surfaces de subdivision est de conserver tous les maillages intermédiaires du processus de subdivision – appelés alors niveaux de résolution – et d'introduire des vecteurs de détail entre ces niveaux. Ces vecteurs expriment la différence en chaque sommet entre sa position au niveau i et celle résultant de la subdivision du niveau $i-1$. Une forte connexion existe entre les surfaces multirésolution et l'analyse en ondelettes, et en particulier les deux opérations classiques dites d'analyse et de synthèse. L'analyse calcule les positions des sommets du maillage de niveau $i-1$ en appliquant un filtre moyenneur sur le maillage de niveau i et calcule par la même occasion les vecteurs de détail. La synthèse reconstruit les données au niveau i en subdivisant le maillage de niveau $i-1$ et en y ajoutant les vecteurs de détail.

Deux méthodes existent pour générer des surfaces de subdivision multirésolution : la méthode "grossier vers fin" et la méthode "fin vers grossier". La première démarre d'un objet grossier (qui constituera le niveau de résolution minimum de l'objet), et construit les niveaux fins en appliquant un schéma de subdivision. La deuxième part d'un objet fin (qui constituera le niveau de résolution maximum de l'objet) et construit les niveaux grossiers en appliquant le processus d'analyse – dans ce cas, le maillage de départ doit avoir une connectivité de subdivision, i.e. la même connectivité que s'il avait été généré par un schéma de subdivision.

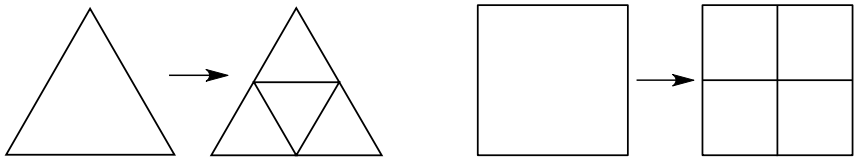
Une fois un tel objet construit, le maillage peut être édité à des niveaux de résolution différents : une édition à un niveau grossier entraîne une déformation globale, en conservant les informations fines grâce aux vecteurs de détail ; une édition à un niveau fin entraîne une déformation plus locale de l'objet. A chaque édition les niveaux plus fins sont mis à jour par le processus de synthèse, et les niveaux plus grossiers par le processus d'analyse.

2.2.2 Schémas de subdivision primaux

Dans un schéma de subdivision primal, ce sont les faces du maillage qui sont subdivisées. Ceci est réalisé en coupant leurs arêtes et en reliant les sommets ainsi créés afin de former de nouvelles faces. Les schémas de Loop [101], de Catmull-Clark [22] ou le schéma Butterfly [51, 140] sont des exemples de schémas primaux. Ceux-ci travaillent sur des maillages de types différents : les schémas de Loop ou de Butterfly sont basés sur des maillages triangulaires (voir figure 2.6a), celui de Catmull-Clark sur des maillages quelconques (voir figure 2.6b), même si après un pas de subdivision toutes les faces deviennent des quadrilatères.

Les surfaces de subdivision multirésolution issues de tous ces schémas peuvent être représentées par des 2-cartes multirésolution. Chaque niveau de résolution est alors représenté par une 2-carte permettant de décrire des maillages quelconques.

Lors d'une étape de subdivision d'un schéma primal, la valence des sommets existants dans



(a) Loop (b) Catmull-Clark

FIGURE 2.6 – Subdivision primale

le maillage initial n'est pas modifiée. Les seules opérations effectuées sont des coupures et des insertions d'arêtes, ces dernières ayant lieu uniquement entre des nouveaux sommets. Cela revient à dire qu'aucun brin n'est inséré dans une permutation α_1 entre deux niveaux, et donc qu'une fois un brin inséré dans la carte à un niveau i, le brin auquel il est lié par α_1 ne change pas aux niveaux supérieurs. Formellement, cela signifie que $\forall x \in N^i, \alpha_1^j(x) = \alpha_1^i(x)$, pour $i < j \leq k$.

Ainsi la topologie d'une surface de subdivision multirésolution peut être modélisée par une 2-carte multirésolution spécialisée en : $M = \left(B, \{\alpha_0^i\}_{i\in[0,k]}, \alpha_1\right)$. Chaque niveau de résolution est alors représenté par la 2-carte $M^i = (B^i, \alpha_0^i, \alpha_1 \mid_{B^i})$ où $\alpha_1 \mid_{B^i}$ est la restriction de α_1 aux éléments de B^i.

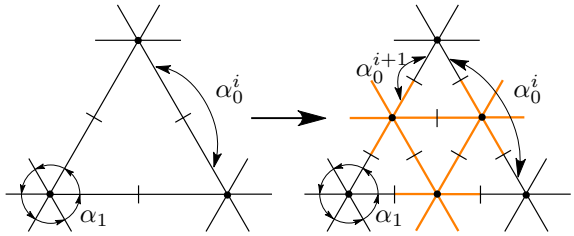


FIGURE 2.7 – Schéma de Loop avec une 2-carte multirésolution

La figure 2.7 illustre un détail de maillage triangulaire aux niveaux i et $i+1$ d'une subdivision de Loop. Les brins en orange sont ceux introduits entre les deux niveaux. La figure 2.8 illustre de la même manière un détail de maillage carré aux niveaux i et $i+1$ d'une subdivision de Catmull-Clark.

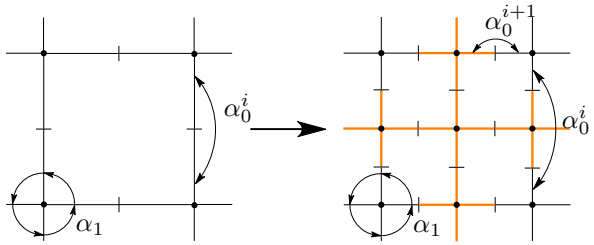


FIGURE 2.8 – Schéma de Catmull-Clark avec une 2-carte multirésolution

L'adaptativité consiste dans les schémas primaux à ne plus subdiviser une face à partir d'un niveau donné. Ceci implique que certaines arêtes du maillage ne sont plus coupées. Formellement, si l'arête du brin x n'est plus coupée à partir du niveau de résolution i, cela signifie que $\alpha_0^j = \alpha_0^i$, pour $i < j \leq k$. La figure 2.9 illustre un détail d'un maillage triangulaire aux niveaux i et $i+1$. Entre ces deux niveaux, seule la face centrale est subdivisée.

Dans la 2-carte représentant le niveau $i+1$, les relations α_0 des brins appartenant à des arêtes n'ayant pas été coupées sont égales à celles du niveau précédent, et les triangles n'ayant pas été

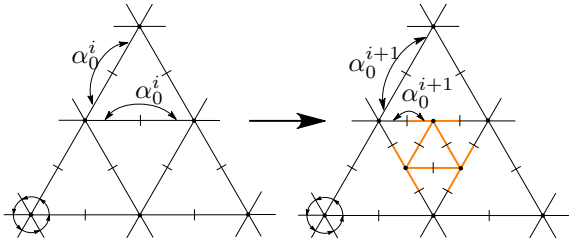
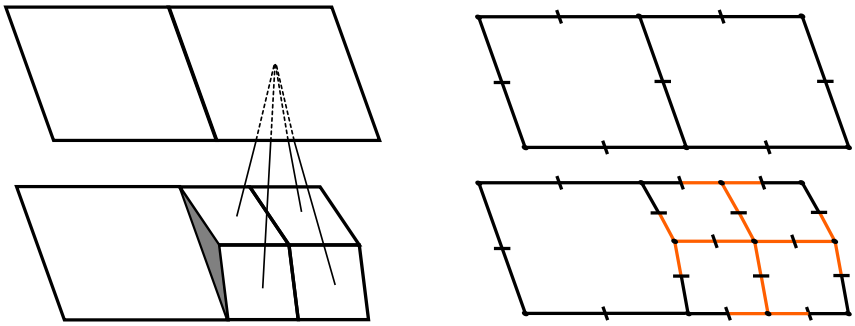


FIGURE 2.9 – Schéma de Loop avec une 2-carte multirésolution adaptative

subdivisés deviennent ici simplement des quadrilatères. Grâce à la représentation implicite des cellules, le maillage est donc ici toujours topologiquement consistant – aucun trou n'apparaît dans la surface.

Classiquement, les surfaces de subdivisions multirésolution sont implantées en utilisant des forêts de quadtrees. Les schémas pouvant être implantés de cette manière sont ceux qui utilisent uniquement la quadrisection de faces ou de sommets, comme ceux de Loop ou Catmull Clark présenté ci-dessus. Chaque face du maillage grossier contient la racine d'un quadtree. Un nœud de cet arbre représente l'état d'une face à un niveau de résolution donné. Les quatre fils d'un nœud représentent les 4 faces résultant d'un pas de subdivision. Les feuilles de l'arbre sont les faces du niveau le plus fin qui ne sont pas subdivisées.



(a) Quadtree (b) 2-carte multirésolution

FIGURE 2.10 – Failles topologique dans un schéma de Catmull-Clark adaptatif

Il faut noter, comme illustré figure 2.10, que dans un quadtree, si deux faces adjacentes ont des niveaux de résolution différents, cela entraine l'apparition de failles topologiques. Ces failles n'existent pas dans les cartes multirésolution grâce à leur capacité à modéliser sans effort supplémentaire des faces de degré supérieur à 4.

2.2.3 Schémas duaux

Dans un schéma de subdivision dual, ce sont les sommets du maillage qui sont subdivisés. Les faces du maillage de départ sont réduites, et les trous ainsi créés sont comblés avec de nouvelles faces. Les schémas de Doo-Sabin [47, 48] (voir figure 2.11) ou le schéma MidEdge [113] sont des exemples de schémas duaux. Ceux-ci travaillent sur des maillages quelconques. Les 2-cartes multirésolution duales peuvent représenter des surfaces de subdivision multirésolution issues de tous ces schémas.

Lors d'une étape de subdivision d'un schéma dual, le nombre de côtés des faces existant dans le maillage initial n'est pas modifié. Les seules opérations effectuées sont des insertions de faces. Cela revient à dire qu'aucun brin n'est inséré dans une permutation ϕ_1 entre deux niveaux, et

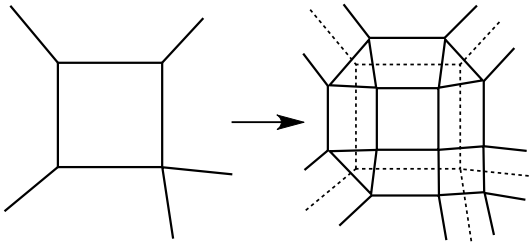


FIGURE 2.11 – Subdivision duale

donc qu'une fois un brin inséré dans la carte à un niveau i, le brin auquel il est lié par ϕ_1 ne change pas aux niveaux supérieurs. Formellement, cela signifie que $\forall x \in N^i, \phi_1^j(x) = \phi_1^i(x)$, pour $i < j \leq k$.

Ainsi la topologie d'une surface de subdivision multirésolution duale peut être modélisée par une 2-carte multirésolution duale spécialisée en : $M = \left(B, \phi_1, \{\phi_2^i\}_{i\in[0,k]}\right)$. Chaque niveau de résolution est représenté par la 2-carte duale $M^i = (B^i, \phi_1 \mid_{B^i}, \phi_2^i)$ où $\phi_1 \mid_{B^i}$ est la restriction de ϕ_1 aux éléments de B^i.

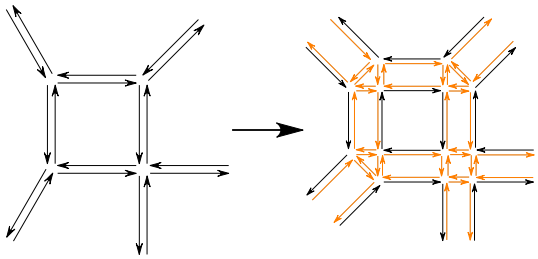


FIGURE 2.12 – Schéma de Doo-Sabin avec une 2-carte multirésolution duale

La figure 2.12 illustre un détail de maillage aux niveaux i et $i+1$ d'une subdivision de Doo-Sabin. Les brins en orange sont ceux introduits entre les deux niveaux – ceux appartenant à N^{i+1}. La figure 2.13 illustre un exemple d'objet obtenu en utilisant le schéma de Doo-Sabin, le maillage en vert est le maillage grossier de départ.

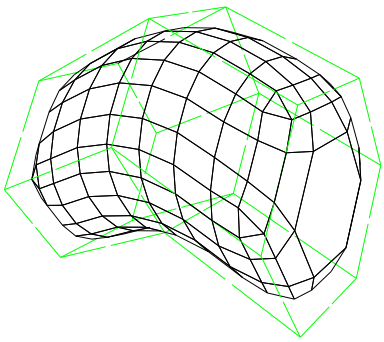


FIGURE 2.13 – Exemple de 2-carte multirésolution duale

2.3 Application aux maillages progressifs

Nous rappelons les notions de base liées aux maillages progressifs et à leur implantation, avant d'indiquer comment ils peuvent être encodés dans des cartes multirésolution et bénéficier ainsi de leurs avantages.

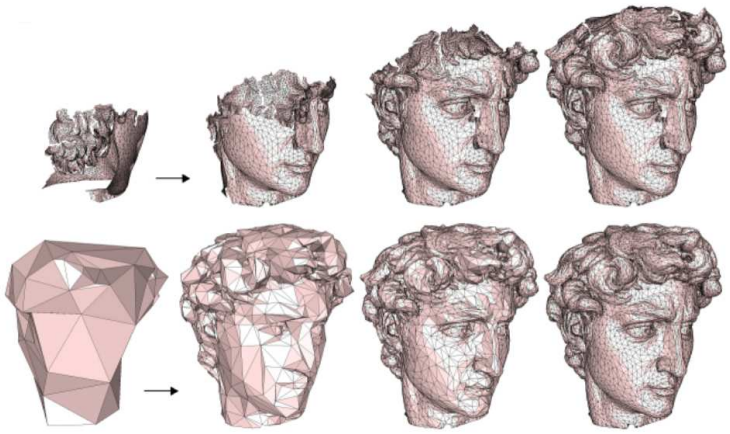


FIGURE 2.14 – Décodage d'un maillage triangulaire compressé avec une méthode directe (en-haut) et progressive (en-bas) (image tirée de [1])

2.3.1 Les maillages progressifs

Les maillages progressifs ont été introduits dans [71]. Ils exploitent deux opérateurs inverses l'un de l'autre : la contraction d'arête et l'éclatement de sommet (voir figure 2.15). Un maillage triangulaire fin M est simplifié par une série de n contractions d'arêtes, jusqu'à obtenir une version grossière M_0. Ce processus produit $n+1$ versions différentes du maillage, notées $M_n \rightarrow M_{n-1} \ldots \rightarrow M_1 \ldots \rightarrow M_0$, chacune étant composée d'un sommet de moins que la précédente. En partant du maillage grossier M_0, il suffit d'appliquer les n opérations d'éclatement de sommet inverses, pour construire la suite de maillage $M_0 \rightarrow M_1 \ldots \rightarrow M_{n-1} \rightarrow M_n$ aboutissant au maillage d'origine. C'est le principe utilisé pour la transmission compressée de maillage (voir figure 2.14).

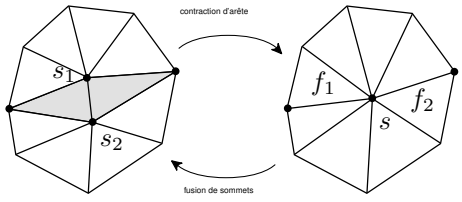


FIGURE 2.15 – Opérateurs de contraction d'arête et d'éclatement de sommet

L'implantation efficace des maillages progressifs est abordée dans [72] et consiste à encoder explicitement le maillage grossier M_0 et les descriptions des n opérations d'éclatement de sommet. Avec les notations de la figure 2.15, ces éclatements sont définis en identifiant le sommet s ainsi que les faces f_1 et f_2 pour savoir où insérer les nouvelles faces. Il faut également pouvoir positionner les sommets s_1 et s_2. Durant la contraction, le sommet s est placé sur s_1, s_2 ou milieu des deux. La donnée d'un vecteur suffit pour recalculer les positions de s_1 et s_2 à partir de celle de s.

Avec cette approche basique, pour obtenir le maillage M_i, il faut systématiquement appliquer les i premières opérations d'éclatement de sommets à partir de M_0. Si cette technique est parfaitement adaptée à la transmission progressive de maillages, elle reste coûteuse pour les algorithmes d'analyse multirésolution qui ont besoin d'un accés aléatoire aux différents M_i.

Une extension à été proposée dans [114], elle consite à organiser les opérations d'éclatement de sommets en fonction de leurs dépendances réciproques. Dans l'exemple précédant les éclatements de s_1 et s_2 dépendent de s, car ils ne peuvent avoir lieu tant qu'ils n'ont pas été créés par l'éclatement de s. Cela conduit à construire une forêt d'arbres binaires (voir figure 2.16) reflétant ces dépendances.

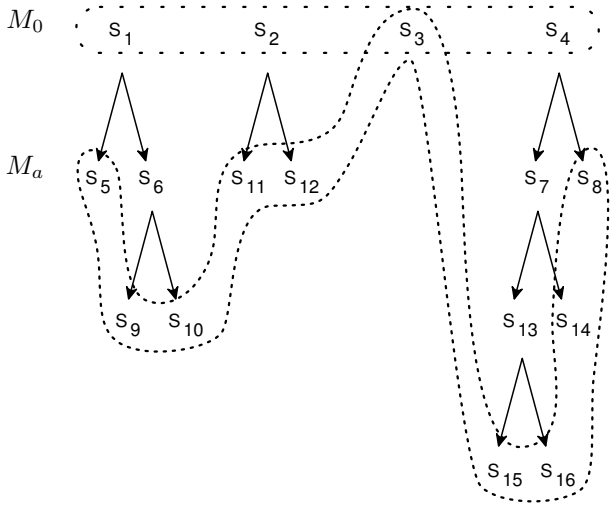


FIGURE 2.16 – Forêt d'arbres binaires de sommets permettant un accès adaptatif dans un maillage progressif. Les pointillés indiquent les opérations effectuées pour obtenir le maillage M_a

En maintenant un *front actif* dans ces arbres, correspondant aux opérations effectuées pour obtenir le maillage courrant, il est possible de naviger plus efficacement dans les différents niveaux de résolution et ce de manière adaptative.

2.3.2 Encodage dans une carte multirésolution

Le modèle des cartes multirésolution permet une représentation des maillages progressifs autorisant un accès direct à toutes les versions intermédiaires du maillage ce qui conduit à une efficacité optimale des algorithmes de traitement de la géométrie exploitant un tel support. Comme il s'agit d'un processus de simplification et non plus de subdivision, la construction des niveaux de résolution se passe ici à l'inverse de ce qui a été montré auparavant.

Chaque étape de simplification consiste à appliquer des contractions sur une sélection d'arêtes dans le maillage courant qui correspond à l'ensemble de brins B^0. Les brins des faces supprimées sont *poussés* dans le niveau de résolution supérieur N^1 avec leur anciennes relations topologiques. Rappelons que les N^i contiennent les brins insérés au niveau i. Durant les étapes suivantes, tous les $N^i (i > 1)$ sont décalés d'un niveau. Ainsi, les ensembles B^i, qui correspondent à l'union des $N^j (j < i)$ se remplissent peu à peu, alors que l'ensemble B_0 des brins du maillage de départ se réduit. Le processus continu jusqu'à obtenir dans B_0 le maillage grossier désiré.

Concernant les relations topologiques entre les brins, elles évoluent comme dans le cas des schémas duaux présentés plus-haut. Les faces ne changent pas d'un niveau de résolution à l'autre. Donc la relation ϕ_1 est la même pour tous les niveaux. La relation ϕ_2 quant à elle reflète les changements d'adjacence entre triangles, comme illustré à la figure 2.18. Dans cette figure sont

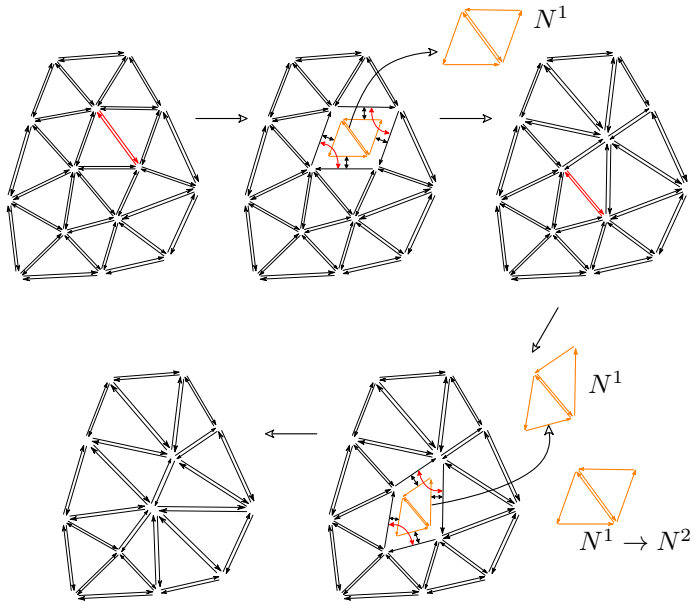


FIGURE 2.17 – Deux étapes de simplification avec les ensembles N^1 et N^2 construits

notées les différentes valeurs de ϕ_2 pour le brin x aux 3 niveaux représentés.

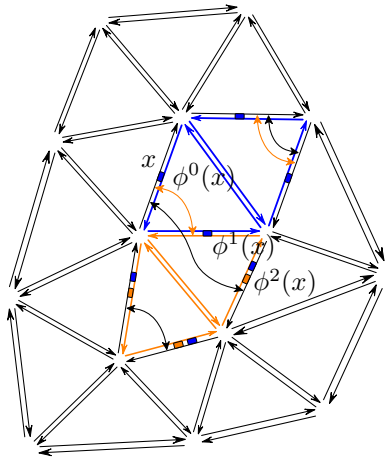


FIGURE 2.18 – Modification des coutures par ϕ_2 au fil des niveaux de résolution

Ainsi la topologie d'un maillage progressif peut être modélisée par une 2-carte multirésolution duale $M = \left(B, \phi_1, \{\phi_2^i\}_{i\in[0,k]}\right)$. Chaque niveau de résolution est directement accessible et représenté par la 2-carte duale $M^i = (B^i, \phi_1 \mid_{B^i}, \phi_2^i)$.

Si les contractions d'arêtes sont effectuées une à une, alors un très grand nombre de niveau de résolution est construit ce qui risque d'être couteux en terme d'occupation mémoire. D'autre part notre but est de générer une hiérarchie de maillages. Pour être exploitable par des techniques d'analyse multirésolution cette hiérarchie doit avoir un nombre restreint de niveaux et chaque niveau doit correspondre à des simplifications réparties de manière homogène sur l'ensemble du maillage.

Pour atteindre ce but, les opérations de contractions sont effectués par paquets de contrac-

tions, indépendantes l'une de l'autre. Une étape de simplification consiste alors à choisir un ensemble d'arêtes disjointes, c'est-à-dire ne provoquant pas la suppression de triangles adjacents, et à les contracter simultanément. Les brins des triangles supprimés sont placés ensemble dans le niveau de résolution supérieur. Ce mécanisme assure d'obtenir des N^i de *bonne* taille (en pratique 20 à 40% des arêtes sont contractées par étape) et donc une hiérarchie exploitable. Cela permet également aux cartes multirésolution obtenues d'occuper moins de place en mémoire comme cela sera démontré plus loin.

2.4 Mise en œuvre et comparaison

Dans cette section nous allons présenter un exemple de structure de données qui implante le modèle des 2-cartes multirésolution que nous venons de définir. Nous nous focalisons ici sur la représentation primale, mais la version duale s'implante aussi simplement.

Nous comparerons ensuite cette structure avec les forêts de quadtrees, classiquement utilisées pour la représentation de surfaces de subdivision multirésolution adaptatives, aussi bien en complexité en temps qu'en besoins en mémoire.

2.4.1 Implantation des cartes multirésolution

Il existe principalement deux manières d'implanter les cartes combinatoires : soit de manière dynamique, avec des listes et des pointeurs, soit de manière plus statique avec des tableaux et des indices. Le conteneur choisi stocke donc l'ensemble des brins de la carte, les relations topologiques étant stockées quant à elles sous forme de pointeurs ou d'indices référençant les brins liés. Si les listes permettent de gérer plus facilement l'ajout ou la suppression de brins, les tableaux offrent des performances plus intéressantes en optimisant les accès mémoire. En dehors de ces nuances, les tailles de structures de données sont du même ordre et les temps d'accès théoriques les mêmes. Par la suite nous parlerons simplement de listes et de pointeurs, en sachant que ces notions peuvent être remplacées par des tableaux et indices sans changer l'analyse qui est faite.

Dans une carte multirésolution il y a autant de relations que de niveaux de résolution. Un brin devrait donc stocker k pointeurs, matérialisant ces relations, pour une carte de niveau maximal k. Heureusement, tous les brins n'ont pas autant de relations. Un brin inséré au niveau l ne possède en fait que $k - l$ relations. Les brins des niveaux grossiers ayant beaucoup de relations sont de fait peu nombreux et les brins des niveaux fins, les plus nombreux, possèdent peu de relations. Au total le coût est amorti sur les différents niveaux de résolutions. Pour permettre cela, il faut distinguer les brins en fonction de leur niveau d'insertion. L'ensemble des brins insérés au niveau i, $B^i \setminus B^{i-1}$ pour $i \in [1, k]$ et B^0 pour $i = 0$, est stocké dans une liste. Les brins de cette liste stockent $k - i$ relations dans un tableau de pointeurs. La carte multirésolution est stockée dans un tableau de telles listes.

L'information géométrique est attachée directement aux brins. Par exemple pour un plongement de sommets, chaque brin contient un pointeur vers un point 3D et tous les brins du même sommet pointent vers le même point. Comme le nombre de plongements dépend également du niveau de résolution, les pointeurs vers le plongement sont également stockés dans un tableau dont la taille dépend du niveau d'insertion du brin.

Ces structures ont été implantées au sein de la plate-forme de modélisation CGoGN [30]. La figure 2.19 illustre des exemples d'objets obtenus par subdivision adaptative avec le schéma de Catmull-Clark. Les objets peuvent être édités à n'importe quel niveau de résolution. L'adaptativité est guidée automatiquement durant l'édition par un critère de courbure : une face n'est subdivisée à un niveau donné que si l'angle formé avec ses faces voisines dépasse un certain seuil ; si l'angle repasse sous le seuil fixé, la face est à nouveau simplifiée.

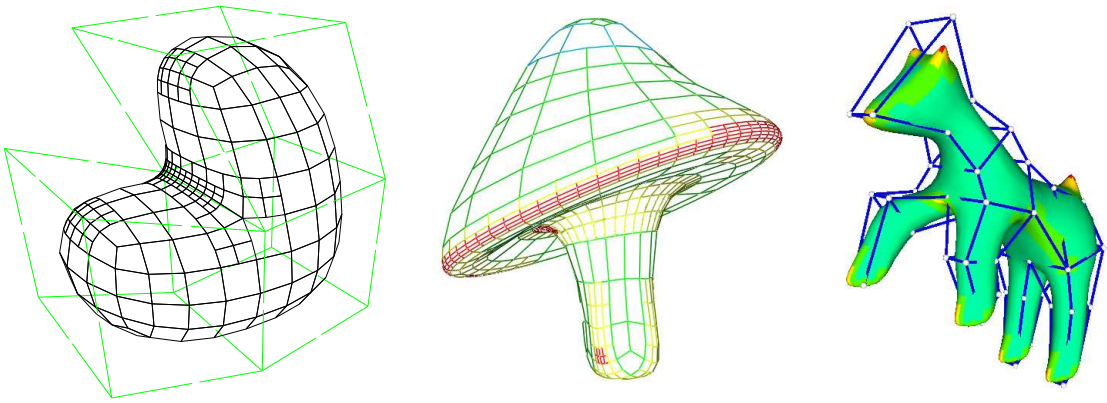


FIGURE 2.19 – Exemple d'application

Nous avons introduit une restriction – non nécessaire dans la définition du modèle – qui interdit à deux faces adjacentes d'avoir plus d'un niveau de différence. Bien que ceci engendre un peu plus de subdivision que nécessaire autour des régions de forte courbure, cela a plusieurs avantages. D'une part, cela permet d'avoir toujours des masques de subdivision pleins, c'est-à-dire qu'un sommet de niveau i dispose toujours de tous ses voisins du niveau $i-1$ nécessaires au calcul de sa position, évitant ainsi le calcul temporaire des positions des sommets qui manqueraient. D'autre part, cela permet de trianguler le maillage obtenu bien plus facilement en vue d'un affichage, en réduisant le nombre de cas à traiter.

2.4.2 Comparaison avec les quadtrees

Complexité en temps

Dans le cadre des surfaces de subdivision, les requêtes d'adjacence entre sommets sont une des opérations les plus courantes. Celles-ci sont nécessaires aussi bien pour l'exécution de l'opérateur de synthèse que pour celui d'analyse.

Dans une carte multirésolution, les requêtes d'adjacence sont exécutées en temps constant, quel que soit le niveau de résolution considéré. En effet, les cellules adjacentes sont récupérées directement en interrogeant les relations stockées pour le niveau i, comme dans une carte normale.

Dans une forêt de quadtrees, les requêtes de voisinage entre faces au sein d'un quadtree sont résolues en remontant dans l'arbre jusqu'au parent commun des deux faces partageant une arête. Le voisinage entre les différents quadtrees de la forêt est résolu explicitement au niveau d'un ensemble de racines qui constitue le niveau 0 de l'objet. Ces requêtes sont exécutées en $O(log(n))$, avec n la profondeur de l'arbre, c'est-à-dire le niveau de résolution maximum. Même si en pratique $log(n)$ reste assez petit, voire borné, sa valeur n'est pas 1. Ces opérations étant les plus utilisées pour la mise à jour de l'objet durant l'édition d'un maillage, cette amélioration n'est pas négligeable.

Complexité en espace

Nous comparons ici le coût mémoire d'une 2-carte multirésolution et celui d'une forêt de quadtrees triangulaires. Comme aucune formulation générale ne peut être donnée dans le cas adaptatif et que la subdivsion régulière est le cas le pire pour les besoins en mémoire, nous effectuons cette comparaison dans le cas régulier.

Pour les cartes : Soit $|B|$ le nombre total de brins d'une 2-carte multirésolution. $|B|$ est égal au nombre de brins nécessaires à la description de l'objet au niveau de résolution maximum. Soit b_0 le nombre de brins décrivant le niveau de résolution 0, et soit k le niveau de résolution maximum. Comme dans les schémas de subdivision primaux présentés ci-dessus le nombre de brins est multiplié par 4 à chaque pas de subdivision, on a :

$$|B| = b_0 \cdot 4^k \tag{2.1}$$

Pour calculer le coût total de l'information topologique, il nous faut compter le nombre de pointeurs stockés par tous les brins. Pour la relation α_1, les choses sont plus simples. Chaque brin dispose d'une unique liaison α_1. Il y a donc en tout $|B|$ ou $b_0 \cdot 4^k$ pointeurs stockés.

Pour la relation α_0, il faut additionner les tailles des tableaux de pointeurs contenus dans chaque brin. La taille de ce tableau est fonction du niveau d'insertion du brin. On note que 3/4 des brins ont été insérés au niveau de résolution maximum et n'ont donc qu'une seule liaison α_0 ; 3/4 des autres brins, i.e. 3/16, ont deux liaisons α_0 ; ... Formellement, pour i entre 1 et k, il y a $|B| \cdot \frac{3}{4^i}$ brins dont le tableau a i éléments. Les brins décrivant le niveau 0 ont $k+1$ éléments dans leur tableau. Le nombre total d'éléments dans les tableaux de liaisons α_0 de tous les brins est donc :

$$b_0 \cdot (k+1) + b_0 \cdot 3 \cdot \sum_{i=1}^{k} i \cdot 4^{k-i}$$

Dans le cas où on utilise un schéma de subdivision approximant, le nombre de plongement dépend du niveau de résolution et correspond en fait au nombre de liaisons α_0. En comptant les pointeurs utilisés pour la géométrie, le nombre total de pointeurs est donc :

$$b_0 \cdot 4^k + 2 \cdot \left(b_0 \cdot (k+1) + b_0 \cdot 3 \cdot \sum_{i=1}^{k} i \cdot 4^{k-i} \right) \tag{2.2}$$

La somme de l'expression (2.2) correspond à la série

$$\sum_{n \geq 0} n \cdot x^n = \frac{x}{(1-x)^2}$$

En négligeant les termes de la série tels que $i > k$ on obtient :

$$\sum_{i=1}^{k} i \cdot 4^{k-i} \simeq 4^k \cdot \frac{\frac{1}{4}}{(1-\frac{1}{4})^2}$$

En simplifiant et en omettant les termes négligeables, on obtient :

$$\frac{33}{9} \cdot b_0 \cdot 4^k \tag{2.3}$$

Pour les quadtrees : cinq pointeurs par nœud sont stockés pour l'information topologique : quatre vers les fils et un vers le père. Les racines des quadtrees stockent sept pointeurs : quatre vers les fils, et trois pour les adjacences entre les racines. Pour l'information géométrique, trois pointeurs vers des points 3D sont stockés dans chaque nœud, ce qui fait donc un total de dix pointeurs par racine, et huit pointeurs pour chacun des autres nœuds. Soit f_0 le nombre de faces du maillage de niveau 0, et k le niveau de résolution maximum. Comme le nombre de faces est multiplié par 4 à chaque pas de subdivision, le nombre total de pointeurs stockés est :

$$10 * f_0 + 8 * f_0 * \sum_{i=1}^{k} 4^i \tag{2.4}$$

La somme de l'expression (2.4) peut être identifiée à la série $\sum_{i=0}^{n} x^i = \frac{x^{n+1}-1}{x-1}$ et être exprimée comme suit (i commençant ici à 1, on retranche le terme tel que $i = 0$) :

$$\sum_{i=1}^{k} 4^i = \frac{4^{k+1}-1}{3} - 1$$

L'expression (2.4) se simplifie alors en (en omettant les termes négligeables) :

$$\frac{32}{3} \cdot f_0 \cdot 4^k \tag{2.5}$$

On peut maintenant calculer le ratio entre le coût d'une 2-carte multirésolution (2.3) et celui d'une forêt de quadtrees (2.5). Dans un maillage triangulaire, il y a trois brins par faces. On a alors $f_0 = \frac{b_0}{3}$. On obtient donc le ratio suivant :

$$\frac{\frac{33}{9} \cdot b_0 \cdot 4^k}{\frac{32}{9} \cdot b_0 \cdot 4^k} = \frac{33}{32} \tag{2.6}$$

On voit que notre modèle ne nécessite environ que 3% d'espace mémoire de plus qu'une forêt de quadtrees. Si on prend en compte le coût de stockage des points 3D, qui est équivalent pour les deux structures, ce ratio est encore plus faible.

Utiliser un schéma de subdivision interpolant rend ce ratio plus avantageux pour les 2-cartes multirésolution que pour les quadtrees. En effet, comme les sommets sont situés sur la surface limite dès leur insertion, l'information de plongement n'a plus besoin d'être paramétrée par le niveau de résolution. Les tableaux de pointeurs de points 3D contenus dans chaque brin se réduisent donc à un seul pointeur. Dans ce cas, le ratio entre 2-cartes multirésolution et quadtrees est de $\frac{30}{32}$. Notre modèle nécessite ici environ 6% d'espace mémoire en moins.

2.4.3 Comparaison avec les maillages progressifs

Le coût mémoire des cartes multirésolution et des maillages progressifs peut être évalué en faisant quelques hypothèse sur les maillages considérés. Nous supposons un maillage sans bord et régulier dont la valence des sommets est de 6, ainsi le nombre de brins de la cartes est simplement égal à 6 fois le nombre de sommets. Nous supposons également un taux de conservation constant, noté t et signifiant qu'à chaque étape de simplification $100.t\%$ des arêtes sont conservés, ou $100.(1-t)\%$ contractées, réduisant d'autant le nombre de sommets. Ce ratio vaut entre 0.6 et 0.8 en pratique. Enfin, nous notons k le nombre de niveaux de résolution de la carte générée. Pour diviser par 100 le nombre de sommets du maillage initial, il faut environ 8 itérations avec un ratio de 0.6 et autour de 20 itérations pour un ratio de 0.8.

Pour l'encodage des maillages progressifs, une carte multirésolution duale dont les ϕ_1 sont fixes est utilisée. Pour estimer le coût des cartes, il suffit de compter le nombre de relations ϕ_2 stockés. Pour le cout des maillages progressifs, ont comptera les ϕ_2 du maillage grossier (les adjacences entre triangles) ainsi que le stockage des opérations de contraction dans des arbres binaires. Les relations ϕ_1, c'est à dire l'ordre des sommets dans chaque triangle, et la géométrie ont le même poids des deux cotés.

Soit N le nombre de sommet du maillage de départ M_0. Durant la première étape de simplification $(1-t).N$ arêtes sont contractées et autant de sommets sont supprimés. Il reste donc $t.N$ sommets dans le maillage. Deux triangles par arêtes sont supprimés, contenant chacun 6 brins. Donc $6.(1-t).N$ brins passent dans le niveau de résolution supérieur. A la fin ces brins seront dans l'ensemble N^k et porteront une seule liaison ϕ_2.

Lors de la seconde étape de simplification, $(1-t).t.N$ arêtes sont contractées. Il reste alors $t^2.N$ sommets. On compte $6.(1-t).t.N$ brins, pour les triangles supprimés, passant dans le niveau

N^{k-1} et portant 2 liaisons ϕ_2. Le coût des liaisons par ϕ_2 au niveau 2 est donc de $6.(1-t).t.N.2$ pointeurs.

Lors de la i-ème étape de simplification, $(1-t).t^{i-1}.N$ arêtes sont contractées. Il reste alors $t^k.N$ sommets. On compte $6.(1-t).t^{i-1}.N$ brins passant dans le niveau N^{k+1-i} et portant i liaisons ϕ_2. Le coût de ces liaisons au niveau i est donc $6.(1-t).N.i.t^{i-1}$. Au final il restera $t^k.N$ sommets dans le maillage de niveau 0 et donc $6.t^k.N$ brins portant $k+1$ liaisons, pour un coût de $6.t^k.N.(k+1)$.

En sommant l'ensemble des coûts calculés, on obtient :

$$\begin{aligned} & 6.t^k.N.(k+1) + \sum_{i=1}^{k} \left(6.(1-t).N.i.t^{i-1}\right) \\ = \quad & 6N\left((k+1)t^k + \sum_{i=1}^{k} i(t^{i-1}-t^i)\right) \end{aligned}$$

Or la somme $\sum_{i=1}^{k} i(t^{i-1}-t^i)$ s'écrit $1-t+2t-2t^2+3t^2-3t^3+\ldots+kt^{k-1}-kt^k$ en simplifiant les termes on obtient $1+t+t^2+t^3+\ldots+t^{k-1}-kt^k$. Le coût devient donc :

$$\begin{aligned} = \quad & 6N\left((k+1)t^k + \sum_{i=0}^{k-1} t^i - kt^k\right) \\ = \quad & 6N\sum_{i=0}^{k} t^i = 6N\frac{1-t^{k+1}}{1-t} \end{aligned}$$

Pour les maillages progressifs, il faut stocker les liaisons du maillage fin contenant $6N$ brins et donc $6N$ liaisons ϕ_2. Puis il faut stocker les opérations effectuées. Dans le maillage final, il reste $t^k.N$ sommets, donc $(1-t^k).N$ contractions ont eu lieu. Chacune stocke des pointeurs vers son père et ses 2 fils, pour la structure d'arbre, et des pointeurs vers les faces adjacentes et le sommet supprimé pour pouvoir être inversée, soit encore 3 pointeurs. Au total, cela donne un coût de $6N(2-t^k)$.

En ajoutant les relations ϕ_1 et les plongements présents dans chaque brin, soit $12N$, le rapport entre le coût des cartes et des maillages progressifs est de :

$$\frac{2-t^k}{1-t^{k+1}}(1-t) \approx 2(1-t) \text{ pour les valeurs de } t \in [0.6, 0.8]$$

Ce ratio est toujours favorable aux maillages progressifs, mais de manière *raisonnable*, avec un surcoût de 25% à 100% pour les cartes. C'est un coût relativement faible en comparaison de celui des données géométriques (coordonnées des points, normales et vecteurs de détails) nécessaires aux traitements géométriques visés.

Les gains en efficacité sont par contre beaucoup plus importants. Dans les cartes multirésolution, l'accès à tous les niveaux de résolution et aux voisinages des sommets dans ces niveaux s'effectue en temps constant ou linéaire en la taille du voisinage. Pour les maillages progressifs, l'accès à un niveau donné implique d'abord l'application des contractions d'arêtes et éclatements de sommets définissant ce niveau. Une fois ce niveau atteint, l'accès aux voisinages est du même ordre que pour les cartes. La plupart des algorithmes de traitement de la géométrie nécessite d'évaluer la position d'un sommet en fonction de son voisinage aux niveaux de résolution inférieur et supérieur. Ce besoin entraine pour les maillages progressifs, de nombreux éclatements et contractions, répétés dans le voisinage des sommets traités. Ces mises à jour incessantes de la topologie sont bien plus coûteuses que les accès directs possibles dans les cartes multirésolution et justifient pleinement le surcout de celles-ci.

2.5 Conclusion

Les cartes multirésolution définies dans ce chapitre offrent un nouveau cadre général pour la représentation de maillages multirésolution. Nous avons démontré deux applications importantes : la représentation de surfaces de subdivision multirésolution et l'encodage de maillages progressifs multirésolution.

Dans le cadre des surfaces de subdivisions, elles apportent un certain nombre d'avantages par rapport aux structures basées sur des quadtrees usuellement utilisées. Le fait de ne pas être limitées aux faces triangulaires ou quadrangulaires, leur permet de représenter, au sein d'un même modèle, des surfaces générées par un grand nombre de schémas de subdivision. La capacité à représenter des maillages polygonaux est un avantage supplémentaire dans le cas adaptatif, où la coexistence de différents niveaux de résolution produit des faces de degré plus important qui ne sont pas supportées par les structures classiques et y engendrent alors des trous topologiques. Les requêtes d'adjacence sont effectuées plus efficacement étant exécutées en temps constant quel que soit le niveau de résolution. Enfin, nous avons montré que ces gains en généricité et en efficacité ce faisait avec un coût mémoire maîtrisé.

Le cadre défini ici amène de nombreuses perspectives dans le domaine des surfaces de subdivision. La généralité des 2-cartes multirésolution permet l'utilisation *au sein d'un même objet* d'algorithmes de subdivision différents. Ceci peut être intéressant dans la mesure où chacun de ces schémas converge vers des surfaces aux propriétés différentes.

En ce qui concerne les maillages progressifs, le principal avantage des cartes multirésolution est qu'elles permettent d'encoder de tels maillages dans une structure réellement multirésolution. Cela permet d'adapter certains algorithmes de filtrages géométriques, habituellement réservés aux maillages obtenus par subdivisions régulières, à des maillages quelconques comme ceux issus de système d'acquisition par scanner.

Nous avons montré comment l'opérateur de contraction d'arête pouvait être utilisé pour construire une hiérarchie homogène de maillages triangulaires. Cette technique pourrait être étendue sans difficulté majeure à d'autres opérateurs de simplification ou à des maillages quelconques supportés naturellement par les cartes.

Enfin, la définition des cartes multirésolution donnée ici en dimension 2 peut sans effort être étendues aux cartes de dimension n. De la même manière, tous les modèles ordonnés cousins des cartes (G-cartes, Hypercartes, X-maps, etc.) peuvent supporter une version multirésolution de leur domaine de représentation, en utilisant le même principe. Le principe consistant à appliquer un opérateur local, de subdivision ou de simplification, puis à transférer les brins modifiés d'un niveau de résolution à l'autre, s'applique à l'identique pour des topologies plus complexes ou de dimensions supérieures.

Chapitre 3

Détection de collisions

Les travaux présentés dans ce chapitre ont été menés pendant la thèse de Thomas Jund [81] encadrée par Jean-François Dufourd et moi-même. Ils ont donné lieu à une publication en revue nationale [78] et à trois communications lors de conférences internationales [77, 80, 79].

Le but de cette thèse était d'explorer une nouvelle approche pour la détection de collisions dans des environnements fortement déformables. Ce type de problématique se retrouve dans de nombreuses applications médicales et notamment la simulation d'endoscopies ou d'opérations chirurgicales micro-invasives où des instruments flexibles sont introduits dans un patient virtuel. La détection de collision est un domaine où beaucoup d'études ont été faites depuis 30 ans. Avec les applications médicales, de nouveaux challenges apparaissent, et notamment la gestion de modèles déformables supportant les changements topologiques (découpes ou sutures par exemple).

La méthode proposée répond à ces nouvelles problématiques tout en restant suffisamment générique pour s'appliquer à d'autres types de simulations. Elle consiste à suivre les déplacements d'un maillage dans une partition convexe de l'espace environnant. La topologie de cette partition est définie sous la forme d'une carte combinatoire. La détection des collisions utilise un système de prédiction mettant à profit la cohérence spatiale des trajectoires et les informations d'adjacence et d'incidence présentes dans les cartes.

Ce chapitre commence par une introduction, section 3.1, où sont exposés le modèle d'animation et la représentation de l'environnement choisis pour notre système de détection de collisions. Le suivi du mouvement d'un maillage est décomposé en deux phases. La première, présentée section 3.3, concerne le déplacement des sommets et leur suivi dans l'environnement. La seconde phase, décrite section 3.5, permet la détection de collisions au niveau des arêtes ou des faces du maillage. Nous présentons également dans cette section les informations de contact pouvant être envoyées au système physique gérant la simulation. Nous discutons des différentes réponses possibles en fonction de l'adaptabilité du modèle.

3.1 Introduction

Un système de simulation interactif est un système complexe, usuellement décomposé en sous-systèmes communicant entre eux. La scène et les mobiles sont modélisés, d'un point de vue mécanique, par un ensemble de particules représentant des masses ponctuelles auxquelles sont associées des informations cinétiques (positions, vitesses). Entre ces particules, existent des interactions mécaniques générant des forces ou des contraintes de natures variées. Celles-ci dépendent des modèles numériques (différences finies, éléments finis, etc.) et physiques utilisées (masses/ressorts, modèles viscoélastiques, *shape-matching*, etc.). Cette information est habituellement stockée dans les cellules d'un maillage reliant les particules. Par exemple, les arêtes peuvent porter la raideur de ressorts, les faces des matrices de raideur et les volumes des positions au repos. Certains modèles dits *meshless* calculent ces informations en intégrant ces données sur un voisinage spatial des particules. Parfois, les actions d'un utilisateur, acquises à partir de périphériques d'interaction, sont intégrées sous forme de forces ou de contraintes s'ajoutant au système.

Dans tous les cas, le sous-système physique calcule, à partir de l'ensemble de ces informations, les forces en présence et les intègre par différentes méthodes numériques. Au final, le système fournit les positions que doivent atteindre les particules ainsi que leurs nouvelles vitesses et directions. En général, on suppose que le temps est suffisamment échantillonné pour considérer des déplacements rectilignes entre deux phases de calculs.

A ce stade le système de détection de collisions entre en jeu. Il doit indiquer si les déplacements prévus sont possibles ou s'ils génèrent des collisions et, le cas échéant, fournir les informations de contact calculées et les nouvelles contraintes du système. Celles-ci peuvent, par exemple, indiquer qu'un mobile glisse le long d'un obstacle ou d'un autre mobile. Les réponses aux collisions sont variables et dépendent des modèles physiques utilisés, mais on peut distinguer deux grandes familles. Dans la première, les déplacements sont réalisés complètement et les interpénétrations causées par les collisions autorisées. Elles sont corrigées, dans les étapes suivantes, par un ensemble de pénalités ajoutées au système qui modifient le comportement mécanique des particules concernées.

Dans la seconde famille de réponses, les mouvements sont bloqués au niveau des points de contacts et les interpénétrations interdites. D'un point de vue mécanique, ces réponses sont adaptées aux corps déformables pour lesquels les tensions internes induites par ces blocages provoquent, après coup, un rebond élastique. D'un point de vue algorithmique l'interdiction des interpénétrations assure que la simulation reste en permanence dans un état fiable et connu. Enfin, en ce qui concerne l'interaction, on assure ainsi que les retours haptiques sont perçus correctement et que les informations visuelles sont conformes à la réalité simulée. Ces deux derniers points sont essentiels pour les applications médicales. Nos travaux se placent donc dans ce cadre de réponses.

3.2 Modélisation de l'environnement

L'environnement dans lequel se déroule la simulation est subdivisé en un ensemble de cellules polyédriques convexes, modélisé par une 3-carte (voir figure 3.1). Il peut correspondre à un réseau vasculaire, à la trachée ou au colon, dans le cas d'applications médicales, mais également à n'importe quelle scène volumique dans laquelle des objets se déplacent. L'utilisation d'une structure topologique forte nous assure que les requêtes de voisinage, fréquentes dans ce qui suit, sont exécutées de manière optimale.

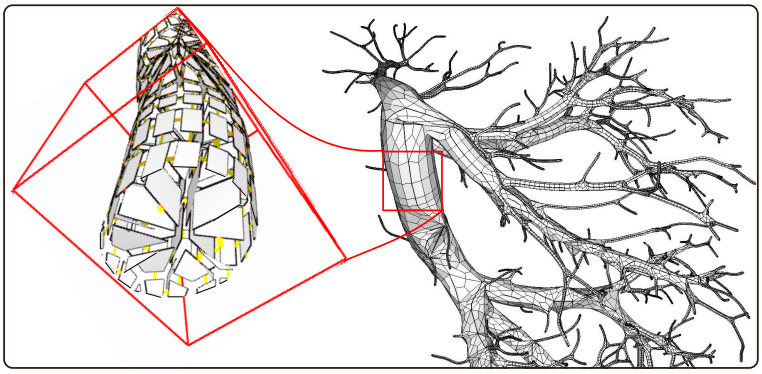


FIGURE 3.1 – Un réseau vasculaire décomposé en cellules convexes.

Habituellement, un tel environnement est représenté en deux niveaux. Le premier est utilisé pour la détection de collisions et le calcul des déformations de l'environnement. C'est souvent un maillage volumique grossier, tétraédrique ou hexaédrique, dont la topologie est fixe. Le second niveau correspond à une surface plus fine, et généralement triangulée, utilisée pour le rendu. Notre approche permet d'unir ces deux niveaux dans une même structure volumique. Les cellules de la carte peuvent être de degrés quelconques, et en particulier les volumes peuvent avoir un bord aussi lisse et découpé que nécessaire pour le rendu. Les changements topologiques sont tous autorisés. On notera que des modèles topologiques similaires ont déjà été utilisés pour traiter des opérations spécifiques telles que la suture ou l'incision dans [10, 100], mais sans lien avec la détection de collisions.

Les mobiles peuvent se déplacer librement dans les cellules de la carte qui sont dites *libres*. A contrario, certaines cellules sont dites *infranchissables*. Elles correspondent au bord extérieur ou à des obstacles internes à l'environnement. Ces obstacles peuvent être des zones volumiques, constitués de plusieurs cellules adjacentes, ou éventuellement de simples faces ou arêtes (voir des sommets) modélisant des tissus interstitiels fins. Des marqueurs, associés aux brins, sont utilisés pour signaler ces zones (figure 3.2(b)).

Il faut noter que la face ou le volume extérieur n'est jamais convexe. Pour que notre méthode fonctionne nous ajoutons des cellules obstacles sur le bord extérieur. En dimension 2, cela correspond à des arêtes pendantes (voir figure 3.2) et, en dimension 3, à des faces pendantes insérées au niveau des arêtes du bord. De cette manière les angles autour des sommets, et des arêtes en en dimension 3, restent aigus ce qui simplifie les tests d'orientation. Il est également possible de considérer un point à l'infini, mais cela sort du cadre de ces travaux.

Les brins d'une carte sont souvent représentés par des demi-arêtes, mais peuvent également être vus comme des n-uplets de cellules. Ainsi, un brin d'une 2-carte représente à la fois un sommet, une arête et une face. Géométriquement, le brin d peut se représenter comme un triangle défini par l'arête de d et un point au centre de la face de d, noté c dans la figure 3.3. Le sommet de d fixe l'orientation de ce triangle. En dimension 3, le brin d peut être représenté par un tétraèdre dont le premier sommet c est au centre du volume, le second au centre c_f de la face et dont

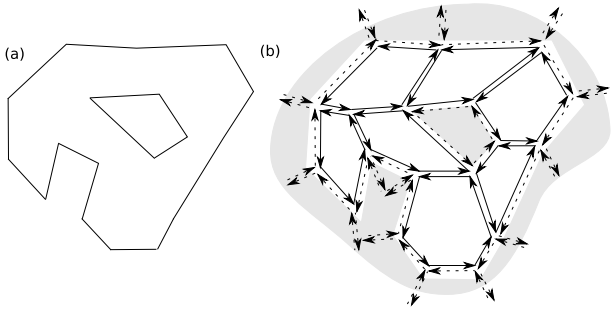


FIGURE 3.2 – Un environnement en dimension 2 et sa décomposition cellulaire ; les cellules infranchissables sont dessinées en pointillés.

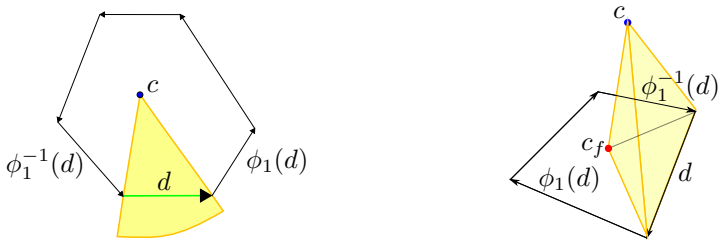


FIGURE 3.3 – Représentation des brins comme des simplexes orientés au sein de leur cellule.

le dernier côté est posé sur l'arête de d. Dans les deux cas, la relation ϕ_1, et son inverse ϕ_1^{-1}, permettent de passer aux triangles ou tétraèdres adjacents, en suivant l'orientation des faces ou l'orientation opposée. En dimension 3, la relation ϕ_2 passe au tétraèdre du *dessus* correspondant à la face adjacente.

Grâce à cette représentation simple, on obtient une décomposition de la 3-carte en un ensemble implicite de tétraèdres. Les cellules étant supposées convexes, cette décomposition est toujours valide et ne contient pas de simplexes dégénérés. Les triangles ou tétraèdres sont des primitives géométriques pour lesquelles tester si un point est inclus, sur le bord ou à l'extérieur est réalisé en 3 ou 4 tests d'orientations simples. L'essence du système de détection de collisions présenté ci-après consiste à savoir à tout moment dans quel simplexe se trouvent les particules en mouvement.

3.3 Suivi des sommets

Le premier étage de notre système de détection de collision consiste à suive le déplacement des particules au sein de la décomposition volumique de l'environnement. Les particules sont placées aux sommets des maillages des mobiles. A chaque pas de temps de la simulation, le système physique fournit la position courante et la position à atteindre. Le suivi des particules est présenté d'abord en dimension 2, pour fixer les principes et notations, avant d'être étendu en dimension 3.

3.3.1 Décomposition des trajectoires

Pour utiliser pleinement la subdivision de l'espace fournie par la carte, en relation avec la cohérence spatiale de l'animation, chaque particule doit savoir dans quelle cellule elle se trouve. En effet, celle-ci a de forte chance d'être dans la même cellule ou dans un voisinage proche, au pas d'animation suivant. Pour permettre une réponse détaillée aux éventuelles collisions la dimension de la cellule est également conservée.

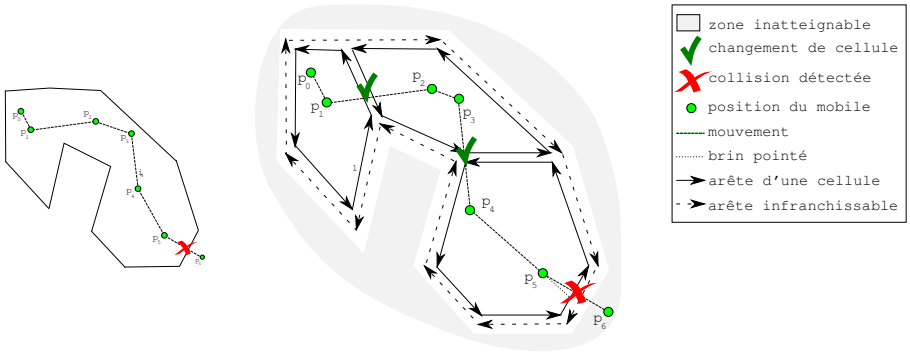


FIGURE 3.4 – Décomposition de trajectoire d'une particule.

Lors du traitement d'un pas de temps, une particule passe de la position p_t à la position p_{t+1}. Pendant ce déplacement, elle franchit un certain nombre de cellules libres, et finit par arriver dans la cellule finale ou, le cas échéant, par rencontrer un obstacle. La trajectoire suivie est ainsi décomposée au niveau des cellules franchies (faces, arêtes ou sommets) comme montré figure 3.4.

Une méthode naïve consiste à tester l'intersection du segment $[p_t, p_{t+1}]$ avec l'ensemble des arêtes de la cellule courante et à en propager les calculs aux cellules voisines lors des franchissements. Le nombre de tests géométriques serait alors égal au degré des cellules traversées ce qui est gênant en dimension 2, et rédhibitoire en dimension 3, surtout lorsque les maillages considérés sont relativement fins. Notre approche consiste à utiliser la décomposition en simplexe présentée plus haut pour suivre pas à pas les cellules traversées, en minimisant le nombre de tests géométriques.

3.3.2 Suivi des particules en dimension 2

Pour exposer la suite, nous ne nous plaçons plus au niveau global d'une trajectoire, mais prenons le point de vue, très local, de la particule. Au début d'un pas de temps, la particule est dans une cellule et se dirige vers une cellule du bord de celle-ci. Par exemple, si la particule se trouve dans une face, elle se dirige vers une de ses arêtes. Lorsque la particule avance sur sa trajectoire, elle se rapproche de l'arête visée et deux cas se présentent. Si elle ne franchit pas l'arête, alors aucune collision ne peut avoir lieu et son état ne change pas, sinon elle change de cellule ou entre en collision avec ce bord.

Pour mettre en place ce schéma d'analyse de manière efficace, chaque particule stocke son état (p_t, k_t, i_t) où p_t est la position de départ, k_t la dimension de la cellule où la particule se trouve (face, arête, sommet), et i_t le brin visé. La particule doit se déplacer vers p_{t+1}. Le point p_t et le brin i_t définissent un triangle élémentaire, comme nous l'avons vu précédemment, que nous nommons par la suite le triangle de *prédiction*.

Intuitivement, l'algorithme consiste à nous assurer que la particule se déplace bien dans le triangle de prédiction, c'est-à-dire que la demi-droite $[p_t, p_{t+1})$ coupe le brin i_t. Puis à tester si la particule franchit le brin i_t ce qui provoque soit un changement de cellule, soit une collision. Après un certain nombre de changements d'états, la particule atteint le point p_{t+1} et vise un nouveau brin i_{t+1} dans une cellule k_{t+1}. Ce nouvel état est utilisé pour l'itération suivante de la simulation, profitant ainsi pleinement de la cohérence temporelle et spatiale des trajectoires.

Nous présentons maintenant, chacun des états de notre mécanisme de prédiction et leurs évolutions possibles. Dans tous les cas, la prédiction est composée d'une phase d'orientation, consistant à valider ou corriger le brin visé i_t, puis d'une phase de déplacement opérant le changement de cellule ou détectant une collision. Dans la suite, nous appelons Δ la demi-droite $[p_t, p_{t+1})$ correspondant à la trajectoire de la particule.

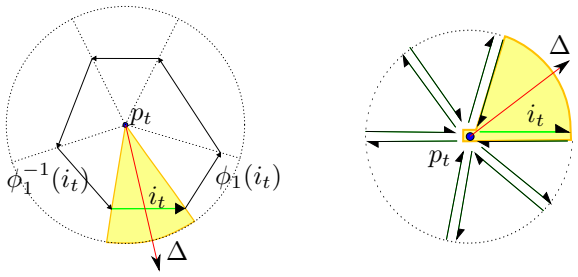


FIGURE 3.5 – Orientation dans une face et dans un sommet. Les brins de la face la partitionnent en secteurs angulaires. Les brins d'un sommet définissent également une suite ordonnée d'angles.

Etat face : la phase d'orientation consiste à tourner dans la face autour du point p_t (figure 3.5), jusqu'à ce que Δ coupe i_t. Si p_{t+1} est à droite de i_t, i_t est remplacé par $\phi_1^{-1}(i_t)$, sinon si p_{t+1} est à gauche de $\phi_1(i_t)$, i_t est remplacé par $\phi_1(i_t)$, sinon p_{t+1} est dans le secteur angulaire représenté en jaune et la prédiction est correcte.

La phase de déplacement, consiste à tester si p_{t+1} est avant i_t ou après (et éventuellement sur i_t). S'il est avant la particule ne franchit pas l'arête et son état ne change pas. L'itération s'arrête sans collision. Dans le cas contraire, si l'arête i_t n'est pas franchissable, une collision a lieu et est reportée mettant fin aux itérations. S'il n'y a pas d'obstacle, la particule change de cellule. Pour propager l'itération aux cellules adjacente, la particule est mise en état arête et le point p_t est placé au point d'intersection de Δ et i_t.

Etat arête : le cas des arêtes est assez simple et se décompose en deux cas. Si p_{t+1} (et donc Δ) pointent vers une des faces adjacentes, on vérifie que cette face est franchissable. Si oui, la particule passe dans l'état face, sinon une collision est reportée. Dans le cas d'un franchissement d'arêtes entre deux faces, cette étape permet de replacer p_t dans la bonne face ce qui valide la décomposition triangulaire utilisée plus loin. Notons que si la particule vient d'une face, ce test d'orientation, entre p_{t+1} et i_t, correspond au test de déplacement de la phase précédente et peut ne pas être recalculé.

Lorsque Δ ne pointe pas vers une des deux faces, c'est que la particule glisse le long de l'arête, vers un de ses sommets. La phase de déplacement teste si le sommet est franchi. S'il n'est pas atteint, l'état de la particule de change pas et les itérations s'arrêtent. Sinon, la particule passe dans l'état sommet et le point p_t est déplacé sur le sommet atteint. Après avoir vérifié que le sommet est franchissable, les itérations se poursuivent ou, le cas échéant, une collision est reportée.

Etat sommet : cet état est le dual de celui de la face (figure 3.5). L'orientation consiste à tourner autour du sommet pour trouver le secteur angulaire contenant Δ. Une fois trouvé, après avoir vérifié que l'arête indiquée est franchissable, la particule passe dans l'état arête. Elle repassera éventuellement dans l'état face après vérification des collisions si Δ visait l'intérieur de la face.

A première vue, des tests plus poussés permettraient de trouver plus rapidement les cellules adjacentes atteintes par la particule. Nous avons choisi de ne pas les faire et de conserver leur simplicité aux traitements des états. Cela permet de garantir que tous les cas sont traités, que les particules n'atteignent jamais un cas non prévu et donc d'assurer la robustesse de l'ensemble.

Complexité : le traitement de ces trois états élémentaires nécessite au maximum trois tests d'orientation point/droite permettant de vérifier si une particule quitte son triangle de prédiction ou pas. Le nombre de triangles parcourus, c'est-à-dire le nombre de fois ou le brin i_t change d'état,

dépend de la longueur du chemin parcouru, du nombre de cellules traversées et de leurs degrés. Précisément, le nombre de tests d'orientation correspond, dans le pire des cas, à 3 fois la somme des degrés des cellules coupées par le segment $[p_t, p_{t+1}]$. En moyenne ce nombre est divisé par deux si on considère que la moitié des brins d'une face (ou d'un sommet) sont parcourus lorsque qu'on la traverse.

Le système que nous avons présenté ici est très général, il permet de considérer des déplacements quelconques dans une partition du plan ce qui assure sa robustesse. Cependant, lors d'une simulation, la longueur des déplacements dépend de la vitesse des particules et surtout du pas d'intégration choisi. En pratique, ces déplacements sont petits, les particules changent donc rarement de cellule et la prédiction, c'est à dire le brin i_t du pas de temps précédent, reste souvent valide. Cela rend cette approche très efficace sur des simulations réelles, comme nous le montrons plus loin.

3.3.3 Suivi des particules en dimension 3

Le suivi des particules en dimension 3 utilise le même principe. Seul le nombre de cas à étudier augmente, puisqu'il faut considérer les volumes. Le triangle de prédiction est remplacé par un tétraèdre, défini par la position de départ p_t, le centre de la face visée (la face du brin i_t), et l'arête i_t. Les tests d'orientation concernent maintenant le placement du point à atteindre p_{t+1} par rapport à ce tétraèdre et comprennent quatre tests point/plan. Les figures étant assez explicites, nous détaillons moins les cas à considérer.

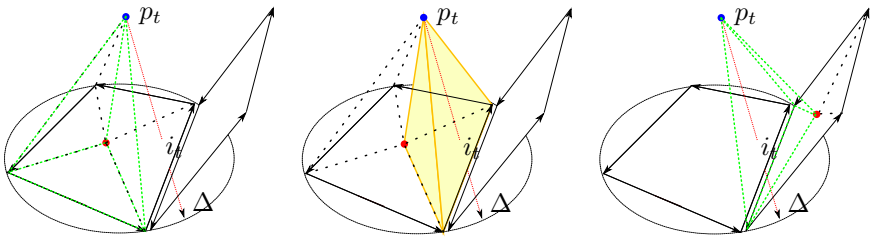


FIGURE 3.6 – Phase d'orientation dans l'état volume ; seule une face adjacente est montrée pour la lisibilité.

Etat volume : la phase d'orientation consiste à tester si le point p_{t+1} est au-dessus ou en dessous des 4 faces du tétraèdre de prédiction. Suivant les cas, le brin i_t tourne dans la face (figure 3.6 gauche) ou passe à la face adjacente (figure 3.6 droite). Après cette orientation la droite Δ traverse le tétraèdre (figure 3.6 au centre). La phase de déplacement consiste à tester si la particule sort du tétraèdre par la face du *fond*. Le cas échéant, il faut vérifier que cette face est franchissable, avant de passer dans l'état face.

Etat face : il s'agit du même cas qu'en dimension 2. Les droites définissant le triangle visé sont remplacées par des plans tangents à la normale de la face $\overrightarrow{n}$. La phase d'orientation permet soit de retourner dans l'état volume, soit de tourner autour de la face, puis d'atteindre l'état arête (voir figure 3.7).

Etat arête : ce cas est similaire au cas des sommets en dimension 2. Les faces incidentes à l'arête définissent des secteurs angulaires. La phase d'orientation consiste à tourner autour de l'arête pour trouver le bon secteur, avant de passer à l'état face. Lorsque le mouvement s'effectue le long de l'arête, on retrouve le cas 2D qui nous amène éventuellement à l'état sommet.

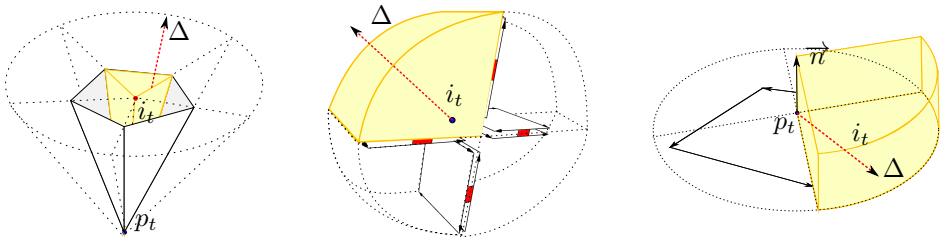


FIGURE 3.7 – Phase d'orientation dans les états sommet, arête et face. Les brins d'une face ou d'une arête définissent des secteurs angulaires ; ceux d'un sommet décomposent son ombrelle en tétraèdres adjacents.

Etat sommet : ce cas est le dual de l'état volume, même s'il est plus difficile à représenter. Le brin i_t d'un sommet appartient à un des volumes incidents à ce sommet. Dans ce volume, le sommet fait partie d'une ombrelle constituée de faces adjacentes du volume (représentées en blanc dans la figure 3.7). Considérons une droite passant par ce sommet et passant à l'intérieur de l'ombrelle. Cette droite permet de décomposer l'ombrelle en tétraèdres (représentés en jaune) qui nous permettent de nous orienter dans le sommet pour trouver le brin correspondant aux volume, face et arête de sortie. Une fois ce brin trouvé, la particule passe par l'état arête, puis éventuellement face et volume, en vérifiant s'ils sont franchissables.

Complexité : comme en dimension 2, le traitement de chaque cas nécessite quatre tests point/plan pour vérifier que le point à atteindre est dans le tétraèdre de prédiction. Si ce n'est pas le cas, l'algorithme passe au brin suivant. Pour un déplacement complet, le nombre de changement d'état correspond, dans le pire des cas, à la somme des degrés des cellules traversées. En pratique, là encore, un nombre réduit de changement est nécessaire. Prenons l'exemple d'un volume ayant de nombreuses faces. Lorsqu'une particule traverse ce volume, les brins parcourus forment un chemin sur le bord du volume. Il est évident que la taille de ce chemin et bien plus petite que le degré du volume, même si une formulation exacte et générale serait difficile à donner. La même remarque s'applique aux sommets. Pour les faces et arêtes, le nombre de brins parcourus est en moyenne la moitié du degré de la cellule.

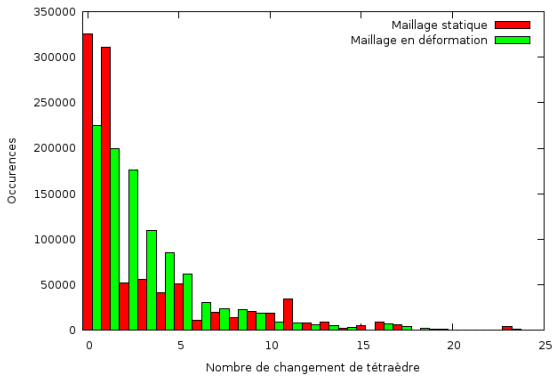


FIGURE 3.8 – Etude statistique du nombre de changements d'état dans une simulation réelle.

La figure 3.8 montre les résultats mesurés pour une simulation réelle. Nous avons lancé quelques milliers de particules dans un maillage volumique. Les particules rebondissaient sur les parois en cas de collision et le maillage était secoué et déformé régulièrement. La simulation a été exécutée assez longtemps pour obtenir quelques millions d'échantillons de déplacements

physiquement cohérents. Le graphe montre le nombre de changements d'états effectués durant les phases d'orientation. On voit que dans quasiment 40% des cas la prédiction était encore valable, dans 40% des cas le tétraèdre adjacent était le bon, et ainsi de suite. Au final, il y a en moyenne uniquement 3 à 4 changements d'état, même lors de déformations fortes du maillage. Cela permet à notre approche d'afficher des performances réelles très intéressantes par rapport aux approches classiques.

Dans ce contexte, un point reste à souligner. La décomposition en tétraèdres utilisée ici pour l'environnement se distingue fortement des décompositions spatiales habituellement utilisées, même tétraédriques. Elle est implicite, car c'est bien une 3-carte qui est utilisée, mais elle est également différente pour chaque particule et pour chaque trajectoire. En effet si la base du tétraèdre est définie par un brin du maillage, qui lui est fixe, le quatrième sommet est toujours défini par p_t, la position atteinte par la particule lors du déplacement précédent. De ce fait, le tétraèdre de prédiction est toujours correctement placé et orienté, et donc adapté par nature à la trajectoire de la particule. Par exemple, lorsqu'une particule traverse un volume en ligne droite, elle utilise le même tétraèdre en permanence. Si ce même volume avait été décomposé en tétraèdre de manière figée, alors le nombre de tétraèdres traversés, et donc de changements d'états, dépendrait de la taille de la subdivision.

Au final, nous avons donc un mécanisme de suivi de particules et de détection de collisions simple, dont la complexité est en pratique indépendante de la taille de la scène et de la subdivision choisie. La donnée importante ici est le rapport entre la taille des cellules et la longueur moyenne des déplacements. Une subdivision volumique grossière est donc suffisante ici et plus efficace qu'une subdivision en tétraèdres ou boites englobantes trop fines.

3.3.4 Réponses aux collisions

En cas de collision, l'état de la particule suivie doit être mis à jour et les informations de contact transmises au simulateur physique. Avec notre approche une partie de la réponse est obligatoire. En effet, les particules doivent toujours être dans un état prévu par le système, elles sont donc systématiquement stoppées aux points de collision. C'est-à-dire que lors d'une collision p_{t+1} est systématiquement placé à l'endroit où elle a lieu.

Les informations retournées sont les suivantes : le temps de la collision, son lieu (cellule topologique et coordonnées du contact), la normale à l'obstacle en ce point, ainsi que la dimension des cellules contenant la particule avant et après la collision. Le temps est estimé avec le ratio de la distance parcourue sur la distance à parcourir. La cellule topologique où a lieu le contact peut être utilisé pour une découpe ou une déformation de l'environnement au point de contact. Les dimensions des cellules sont transmises car elles ont une signification physique. Par exemple le passage, pour une particule, d'un volume à une face implique un choc avec la face et éventuellement le calcul d'une force de rebond. Une particule partant d'une face dont le déplacement la laisse sur une face (éventuellement adjacente) implique un glissement et non un choc et donc éventuellement le calcul de forces de frottement.

Le suivi de particules proposé est suffisamment général pour s'adapter à tout type de modèle physique, dès lors que la condition de blocage est acceptable par ce modèle.

3.4 Déformations et modifications de l'environnement

Dans la section précédente, nous avons supposé, implicitement, que les cellules de l'environnement n'étaient pas déformées et leur topologie non modifiée. Il convient maintenant de vérifier les incidences qu'ont ces éventuels changements sur notre système de prédiction et de suivi. On peut considérer, en restant générique, que le déplacement des sommets de l'environnement et les changements topologiques sont effectués après le déplacement des particules, ou du moins successivement, et non de manière concurrente.

3.4.1 Déformations des cellules

Ainsi, considérons une particule, placée dans la cellule du brin visé i_t, ayant effectué le trajet $[p_t, p_{t+1}]$, où p_t est l'avant dernier point atteint durant les itérations et donc appartient à la cellule courante. On sait donc que p_{t+1} est dans le triangle ou le tétraèdre de prédiction défini par p_t et i_t. Lorsque la cellule de i_t est déformée, trois cas peuvent survenir, illustrés dans la figure 3.9

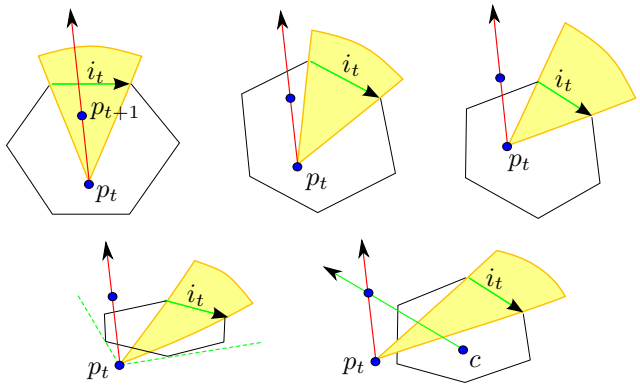


FIGURE 3.9 – Positions relatives de p_t, i_t et p_{t+1} lors de déformations du maillage. La première sous figure (haut gauche) correspond à la prédiction avant déformation. Les autres sous figures représentent les différents cas à traiter. Pour éviter de les surcharger, seule la demi-droite Δ est représentée.

Prédiction toujours valide : p_t et p_{t+1} restent dans la cellule i_t et p_{t+1} est toujours dans le tétraèdre de prédiction. Il n'y a ici rien à faire.

Orientation modifiée : p_t et p_{t+1} restent dans la cellule i_t, mais p_{t+1} n'est plus dans le triangle ou tétraèdre de prédiction. Il suffit de refaire une phase d'orientation, c'est-à-dire de simuler un nouveau déplacement de p_t vers p_{t+1}, pour mettre à jour i_t.

Cellule franchie : p_t est encore dans la cellule i_t, mais p_{t+1} ni est plus, la prédiction est alors invalide. Ce cas survient lorsque le bord de la cellule traverse p_{t+1}. Il faut trouver la nouvelle cellule contenant p_{t+1}. Comme p_t et i_t sont encore valide, il suffit là encore de simuler un nouveau déplacement de p_t vers p_{t+1}, pour corriger i_t et la prédiction.

Cellule totalement franchie : p_t et p_{t+1} ont quitté la cellule i_t ou, autrement dit, celle-ci a été suffisamment déformée pour que son bord traverse les deux points. Nous décomposons ce cas en deux sous cas :

- Si p_{t+1} appartient au cône formé par p_t et la cellule i_t, alors la phase d'orientation est possible. Durant la phase de déplacement, p_t sera replacé sur le bord de la cellule franchie et les itérations suivantes seront valides.
- Dans le cas contraire, la phase d'orientation boucle sur elle-même, car p_{t+1} sera à droite (ou à gauche) de tous les brins de la cellule. Ce cas est facilement identifiable, en conservant le brin de départ dans la boucle d'orientation. Pour rétablir les choses, il faut remettre des valeurs correctes dans l'état de la particule. Le point p_{t+1} ne peut être modifié, c'est le point de départ pour le prochain pas de simulation. La cellule contenant p_{t+1} est celle que l'on cherche. Il faut donc partir de la seule donnée connue i_t et corriger la valeur de p_t. Pour cela il suffit de choisir un point de la cellule i_t, par exemple son barycentre, et simuler un déplacement de ce point vers p_{t+1} pour recouvrer toutes les informations.

Une remarque s'impose ici. Pour détecter dans quel cas, on se trouve, il faut tester si p_t et p_{t+1} sont encore dans la cellule i_t. Cela reviendrait naïvement à tester l'ensemble des brins du bord, ce qui serait trop coûteux. En pratique, il suffit de relancer dans tous les cas une phase

de déplacement de p_t vers p_{t+1}. Si l'orientation était correcte (comme dans le premier cas), ce déplacement s'arrête dès la première itération, sinon l'orientation et la cellule i_t sont corrigées (cas deux et trois). Si la phase d'orientation boucle, p_t est placé au barycentre de la cellule et le déplacement relancé.

Au final, ce mécanisme permet de gérer les déformations du maillage environnant, en dédoublant simplement le déplacement de particules, c'est à dire avec un effort minimal en comparaison de la mise à jour d'une hiérarchie de boites englobantes. Le diagramme de la section précédente (figure 3.8) montre bien que le surcoût d'un maillage déformable est très faible. Le nombre moyen de changements d'états n'est en pratique pas multiplié par deux, mais par un facteur plus faible. En effet, les ajustements d'orientation effectués lors du déplacement des particules et lors de la déformation de l'environnement se complètent et s'ajustent mutuellement.

3.4.2 Changements topologiques et remaillage convexe

La gestion des changements topologiques est tout aussi simple que celle des déformations. Considérons une particule, son ancien déplacement et sa prédiction définis par p_t, p_{t+1} et i_t. Lorsqu'une opération topologique est effectuée sur la cellule de i_t, il suffit de vérifier que la prédiction reste valable. Si la cellule est agrandie, par exemple en la fusionnant avec une autre ou en subdivisant son bord, alors les deux points p_t et p_{t+1} restent dans la cellule. L'orientation vers i_t ne changeant pas en l'absence de déplacement, la prédiction reste valable.

Si la cellule est subdivisée ou découpée, cela signifie que des cellules de dimensions plus petites ont été ajoutées à l'intérieur de la cellule i_t, comme une arête (respectivement une face) découpant une face (respectivement un volume). La prédiction doit être mise à jour si ces cellules traversent le triangle ou le tétraèdre de prédiction, c'est à dire passent entre p_t et i_t. L'ajout de cellules entre p_t et i_t implique en fait que p_t ne se situe plus dans la cellule de i_t, mais dans la nouvelle cellule issue de la découpe. Par contre, parmi les brins ajoutés, ceux qui forment le bord cette nouvelle cellule contenant p_t sont connus. Il suffit de remplacer i_t par l'un de ces brins. Pour finaliser la correction, un déplacement de p_t vers p_{t+1} est effectué. Cela permet de corriger l'orientation et de trouver parmi les brins ajoutés celui qui portera la prédiction ou, si p_{t+1} était de l'autre coté des cellules ajoutées, de replacer p_t et i_t dans la nouvelle cellule crée.

Pour être complet, il faut considérer la possibilité de la suppression d'une cellule, par un opérateur de contraction ou parce qu'elle est devenue dégénérée. La prédiction n'est dans ce cas plus valide, mais peut être corrigée simplement en plaçant i_t dans une cellule adjacente et en replaçant p_t au centre de cette cellule, puis en effectuant à nouveau le déplacement vers p_t+1.

Comme pour le traitement des déformations ce mécanisme est simple et efficace. La correction des changements topologiques nécessite uniquement des corrections locales bien identifiées ce qui montre la pertinence de notre approche pour la gestion d'environnements de simulation fortement déformables. Ces changements peuvent être induits par la simulation (découpe ou suture de l'environnement), mais peuvent également intervenir dans le cadre du maintien de la convexité des cellules. Lorsque les déformations sont importantes, il est en effet possible de perdre localement la convexité d'une cellule. Cette perte doit être détectée et corrigée pour que le système fonctionne. L'algorithme de remaillage en lui même sort du contexte de ces travaux, mais l'analyse proposée précédemment pour la gestion des déformations et des changements topologiques montre qu'il est possible de remailler localement l'environnement tout en retrouvant des prédictions valides, pour un coût algorithmique très réduit.

3.4.3 Réponses aux collisions dues à l'environnement

Quelques soient les modifications apportées au maillage modélisant l'environnement, nous avons vu qu'un second lancer de particule permettait de corriger les informations de prédiction. Durant ce déplacement factice, il est possible de découvrir une collision. Cela arrive, par exemple,

lorsque qu'un obstacle repousse le mobile. Deux réponses sont possibles, le choix dépendant du modèle physique et du comportement souhaité.

La première consiste simplement à déplacer la particule au point d'impact. Physiquement, la particule se retrouve en contact avec l'obstacle, dont elle ne devait pas être éloignée auparavant. Sa vitesse devrait être modifiée en conséquence. La seconde méthode consiste à bloquer le mouvement de l'obstacle, pour l'empêcher de passer au travers du mobile. Ce comportement convient mieux pour des mobiles dont le mouvement est contrôler par l'utilisateur. Il convient de générer une force de répulsion au niveau de la particule bloquante. Pour les obstacles qui sont des corps déformables, ces blocages induisent des tensions internes qui provoqueront d'autres déformations.

Dans tous les cas, comme pour le déplacement des particules, il faut s'assurer de la stabilité du système. Après le traitement choisi, les particules doivent être placées dans ces cellules franchissables de l'environnement, bien identifiées. Dans une simulation de corps déformables en contact, on ne cherche d'ailleurs pas à savoir qui de l'obstacle ou de la particule pousse l'autre, mais plutôt à identifier précisément les interactions pour pouvoir générer les contraintes et forces mécaniques adéquates. L'approche présentée ici fournit ces indications de manière robuste et précise.

3.4.4 Robustesse et approximations numériques

L'ensemble du système de suivi présenté dans les sections précédentes est basé sur l'évaluation d'un unique prédicat qui correspond à la classification d'un point par rapport à une droite, en dimension 2, ou à un plan, en dimension 3. Le système est stable ou robuste si les particules sont toujours dans un état identifié et prévu par le système. Pour cela quelques précautions doivent être prises. Une particule doit toujours appartenir à la cellule la plus contraignante ou la plus petite. Par exemple, une particule sur une arête, peut être également considérée comme étant dans la face incidente. La définition du prédicat doit interdire cette possibilité.

Si une programmation rigoureuse des prédicats doit permettre d'éviter la plupart des problèmes d'approximation numériques, ceux-ci arrivent tout de même. Ainsi, par exemple pour l'état volume, si les points p_t, p_{t+1} et le centre de la face visée sont parfaitement alignés, la phase d'orientation peut boucler. D'autres configurations, mêlées à des erreurs d'arrondis, peuvent conduire à des boucles infinies lors des changements d'états. Pour détecter ces cas, il suffit de mettre en place un système d'horodatage des brins parcourus, chaque suivi incrémentant l'heure de parcours. Si lors d'un suivi, on passe par un brin visité à la *même heure*, alors une boucle est détectée. Pour réinitialiser la particule fautive, il suffit de repartir d'une cellule connue (la cellule du pas précédent), en initialisant le point p_t avec une autre valeur, comme par exemple le centre de la cellule.

Nous avons constaté, en pratique que ces techniques simples à mettre en oeuvre, faisaient disparaître les problèmes numériques. Des techniques plus avancées, comme des arithmétiques d'intervalles, pourraient être utilisées en cas de besoin et pour des simulations spécifiques.

3.5 Suivi d'un maillage

Dans cette section, nous montrons comment, le suivi des particules présenté auparavant nous permet de gérer les collisions d'un maillage avec son environnement.

3.5.1 Convexité et déplacements élémentaires

Lorsque l'on déplace un maillage complet, on peut considérer que ses sommets sont déplacés tour à tour. Le déplacement d'un sommet induit le déplacement des arêtes et éléments de surfaces qui l'entourent. Les surfaces ou les volumes balayés par ces déplacements sont composés de

triangles (pour les arêtes) ou de tétraèdres (pour les faces) que nous appelons des déplacements élémentaires. Les figures 3.10 et 3.11 illustrent ces notions.

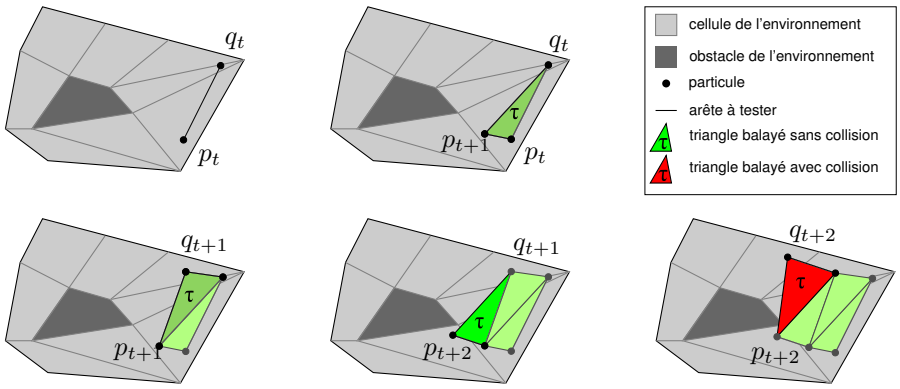


FIGURE 3.10 – Déplacement d'une courbe dans le plan et triangles élémentaires.

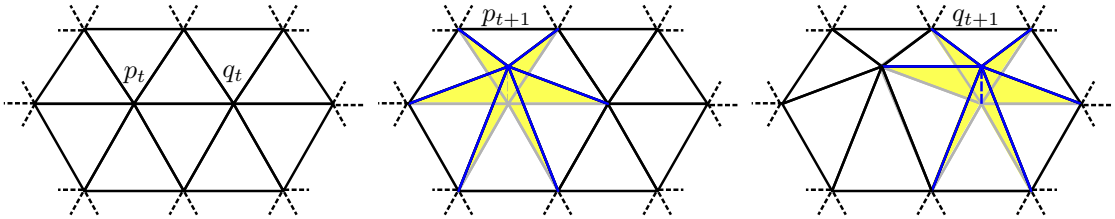


FIGURE 3.11 – Déplacement d'une surface dans l'espace, avec les triangles élémentaires et les tétraèdres correspondant (non surlignés).

Les déplacements des sommets s'effectuent dans une partition convexe, ainsi quelques propriétés géométriques simples peuvent donner de bonnes indications sur les intersections des déplacements élémentaires avec l'environnement.

Considérons le cas d'un triangle. Si ses trois sommets sont dans la même cellule, alors le triangle est complètement inclus dans cette cellule et ne peut donc pas contenir d'obstacle. Si ses sommets sont dans des cellules distinctes, alors un obstacle inclus dans le triangle est soit complètement inclus dans celui-ci, ou bien traverse l'une de ses arêtes.

Ces propriétés s'étendent aux tétraèdres. Si ses quatre sommets sont dans une même cellule, un tétraèdre ne contient aucun obstacle. Si ses sommets sont dans des cellules distinctes, alors un obstacle est soit complètement inclus dans le tétraèdre, soit traverse l'une de ses faces. Dans ce dernier cas, soit l'obstacle est assez fin pour traverser uniquement l'intérieur de la face, soit il traverse également une de ses arêtes.

Notons tout de suite ici que, dans le cadre d'une simulation physique, les déplacements sont petits. En conséquence les hauteurs des triangles et tétraèdres correspondant à des déplacements élémentaires sont faibles en proportion de leurs bases. Les configurations où un obstacle est complètement inclus dans ces primitives élémentaires ou les traverse sans passer par leurs arêtes, correspondent à des obstacles soit très petits, soit très fins et donc pointus. Ceux sont des cas de figures rares en simulation qui nous amène à considérer ces cas comme improbables. Nous proposons donc des optimisations permettant des les ignorer. Le traitement complet et toutefois possible, mais à des coûts un peu plus élevés.

3.5.2 Suivi des arêtes

Nous considérons tout d'abord le déplacement de courbes dans une partition du plan où les obstacles sont plus grands que le triangle balayé par un déplacement élémentaire. Ce triangle, noté τ par la suite, a pour sommets $\{p_t, p_{t+1}, q_t\}$. Si les trois sommets de τ sont dans la même cellule de l'environnement, alors il ne contient pas d'obstacle et les calculs s'arrêtent ici. Notons que dans le cadre d'expérimentations sur des simulations réelles, nous avons constaté que nous étions dans cette configuration dans quasiment 30% des cas. Il n'est pas rare, en effet, que les mobiles ou outils en déplacement soient plus subdivisés que l'environnement.

Dans le reste des cas, il faut vérifier que τ ne contient pas d'obstacle, et pour cela, s'assurer qu'aucun obstacle ne coupe ses arêtes. La figure 3.10 décompose le déplacement d'une arête dans ce cadre. Le triangle τ correspond au déplacement de p_t vers p_{t+1}. L'arête $\{p_t, q_t\}$ a été vérifiée au pas de temps précédent puisqu'elle correspond à un déplacement déjà effectué. L'arête $\{p_t, p_{t+1}\}$ est vérifiée lors du déplacement de la particule p_t vers p_{t+1}. Celle-ci a éventuellement déjà été bloquée en cas de collision. Dans tous les, il n'y pas d'obstacle entre p_t et p_{t+1}.

Il reste la troisième arête à vérifier. Cela est fait en simulant le déplacement d'une particule de p_{t+1} vers q_t. Pour une courbe ou un maillage quelconque, ce lancer de particule est réalisé pour toutes les arêtes adjacentes au sommet de p_t. Si une collision est détectée, alors un obstacle traverse le triangle τ. Si le temps d'exécution de la détection de collisions doit être limité strictement on peut se contenter de cette réponse binaire et bloquer le déplacement de p_t vers p_{t+1}. Sinon il est possible d'obtenir des informations plus détaillées sur la collision.

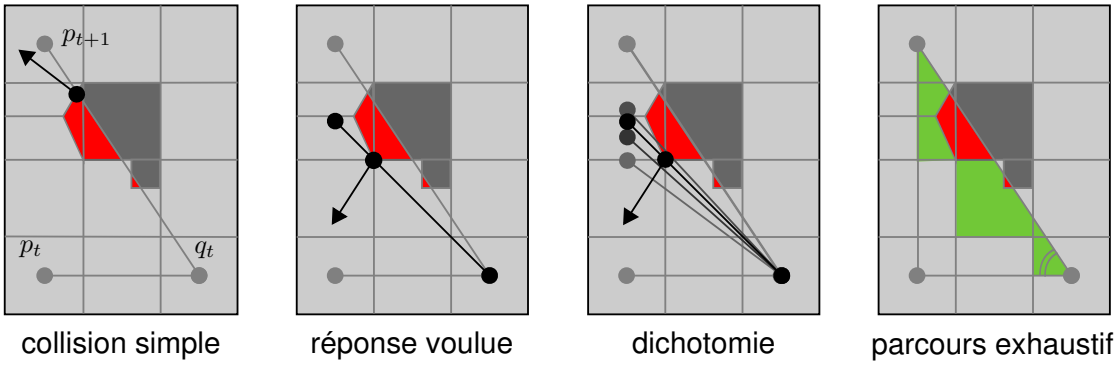


FIGURE 3.12 – Traitement des collisions d'arêtes dans le plan.

La figure 3.12 décrit les informations que l'on peut obtenir en cas de collision d'arêtes et comment elles sont calculées. Illustré à gauche, le déplacement de p_{t+1} vers q_t ne fournit qu'une information partielle. En particulier, le point de contact ne correspond pas au premier point de l'obstacle rencontré par l'arête. La seconde figure montre ce que devrait être une réponse précise et correcte.

La première méthode, simple à mettre en oeuvre, consiste à effectuer une recherche dichotomique le long du segment $[p_t, p_{t+1}]$, jusqu'à trouver un point $p_{t+\alpha}$ tel qu'une particule puisse aller de $p_{t+\alpha}$ vers q_t, sans obstacle. Ce point est une approximation de la réponse qui laisse le système dans un état correct. En pratique, 1 ou 2 pas de dichotomie sont largement suffisants. Si on se souvient que, durant le même pas de temps de la simulation, la particule q_t doit également bouger entraînant vraisemblablement une collision d'arête au même endroit, c'est même sans doute la meilleure réponse que l'on peut donner, d'un point de vue physique. Mécaniquement cela signifie que l'arête est freinée en arrivant sur l'obstacle, son mouvement n'étant pas complet. Ceci entraîne l'apparition de tensions internes dans le mobile et provoquera le rebond souhaité.

La seconde méthode est plutôt adaptée à la simulation de bras articulé ou de solides non déformables. Elle consiste à faire une recherche exhaustive le long de l'arête $\{p_{t+1}, q_t\}$. Pour cela,

le déplacement de la particule de p_{t+1} vers q_t est effectué sans interruption, même en cas de collisions, afin de parcourir toutes les cellules comprises entre les deux points. A partir de cette liste, il est simple de trouver le sommet de l'obstacle minimisant l'angle avec $\{p_t, q_t\}$. Le temps de calcul dépend ici du nombre de cellules à traverser. Par contre la réponse fournie est exacte.

Dans tous les cas, notons que toutes les arêtes incidentes au sommet déplacé doivent être examinées avant son déplacement effectif. Concrêtement, on note pour chaque arête le temps de collision trouvé, puis le sommet est avancé jusqu'au point correspondant à la première collision. Dans ce cas, la dichotomie peut s'effectuer plus efficacement sur l'ensemble des arêtes du sommet, en n'itérant que sur les arêtes encore en collision.

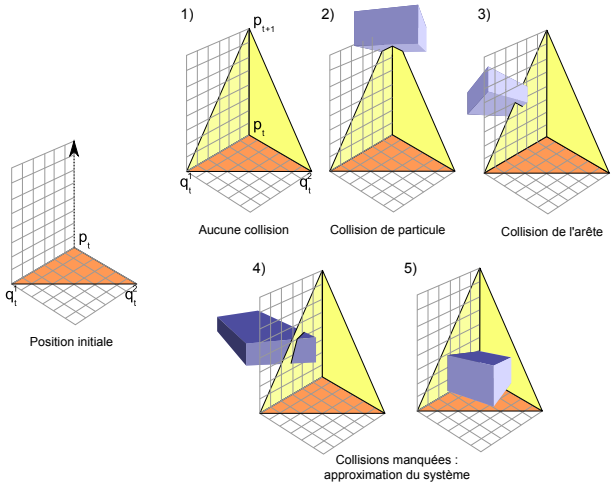


FIGURE 3.13 – Détection de collisions limitée aux arêtes en dimension 3.

La détection de collisions le long des arêtes du mobile fonctionne exactement de la même manière en dimension 3. La nature des obstacles qui ne peuvent pas être détectés change un peu, comme illustré figure 3.13. Les obstacles complètement inclus dans un tétraèdre formé par le déplacement d'une particule peuvent être encore une fois ignorés, ils sont beaucoup trop petits pour correspondre à une simulation réelle. Par contre, les bords pointus des obstacles peuvent être manqués (cas 4) et traverser les faces du maillage. A contrario, rappelons que le mobile lui, ne peut jamais traverser les cellules infranchissables de l'environnement. L'approximation évoquée ici ne les concerne pas. Si celle-ci n'est pas compatible avec les besoins de la simulation, l'approche exacte présentée ci-dessous doit être utilisée.

3.5.3 Suivi des faces

Considérons à nouveau le triangle τ. Ces trois arêtes correspondent à des déplacements de particules réalisés complètement. Si on stocke, pour chacun d'entre eux les brins traversés lors de ce parcours, on obtient tous les brins à la frontière de τ sans calcul supplémentaire. Différentes configurations sont montrées figure 3.14. Un parcours de la carte, partant de ces brins et faisant avancer un front vers l'intérieur de τ en parcourant les brins adjacents, permet de vérifier de manière exacte l'existence d'obstacles dans ce triangle.

Un peu plus complexe à mettre en oeuvre, cette méthode reste assez peu coûteuse en pratique. Si on considère que les arêtes du mobile et les déplacements sont plus petits ou de tailles comparables à celles des cellules de l'environnement, alors, en moyenne, seul un brin par arête doit être parcouru. En général, le graphe à l'intérieur de τ se limite à un sommet. Si une des arêtes passe par un sommet de l'environnement, le nombre de cellules à vérifier peut être un peu plus élevé.

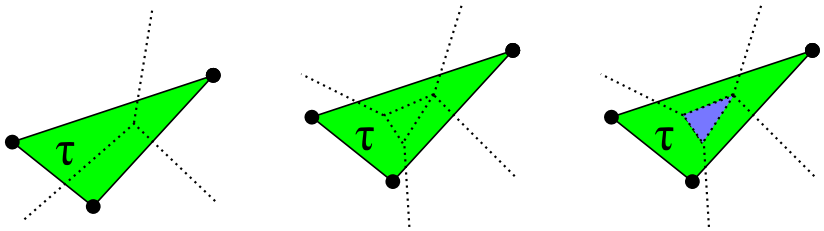


FIGURE 3.14 – Brins traversant un triangle de balayage élémentaire, sans obstacle (gauche et centre) ou avec obstacle (droite).

Avec cette méthode qui nécessite de stocker l'historique des brins franchis par les derniers suivis de particules, on sait avec certitude si les faces engendrées par le déplacement du mobile traverse un obstacle. Le cas 4 de la figure 3.13 est donc évité. Par contre, le cas 5 ne l'est pas, s'il existe des tétraèdres suffisamment petits pour être complètement inclus dans un tétraèdre de balayage.

Le principe évoqué plus haut pour le suivi des faces peut être étendu aux volumes. Pendant le parcours du graphe correspondant à l'intérieur de τ, il est encore possible de stocker les brins atteints et donc traversés par la faces. En partant des brins stockés pour les quatre faces du tétraèdre, avec un parcours de graphe similaire, il est possible de vérifier complètement l'intérieur du tétraèdre. Nous n'avons pas encore trouvé d'applications nécessitant le déploiement de tels efforts, mais le principe reste général et adaptable à d'autres situations.

3.5.4 Réponses aux collisions d'arêtes ou de faces

Les collisions, au niveau des arêtes ou des faces, sont détectées en examinant les déplacements élémentaires générés lorsqu'elles balayent l'espace. L'ordre dans lequel ces déplacements sont évalués, influence bien sur la position de la collision, mais également la réponse générée et son lieu d'action. Par exemple, si l'arête d'un mobile doit entrer en contact avec un obstacle, le déplacement d'un des sommets avant l'autre induit la découverte d'une collision différente, comme illustré figure 3.15 (cas a et b). Les forces calculées et donc la réponse mécanique sont alors disymétriques. Cet aspect est lié au caractère local du mécanisme de détection proposé, qui en fait par ailleurs toute son efficacité. Il y a deux moyens simples de pallier à ces inconvénients

Le premier consisterait à évaluer l'existence de collisions dans un voisinage direct des cellules du mobile, pour rechercher l'impact le plus vraisemblable. Il s'agirait, par exemple pour une arête, de simuler le déplacement succéssif de ses sommets et de calculer le temps minimal avant collision. Au départ, dans les cas a et b, p_t (ou q_t) est avancé jusqu'à p_α (ou q_β) et q_t (ou p_t) n'est plus déplaçable, ce temps est donc nul. En itérant par dochotomie sur les positions intermédiaires de p_α et q_β, on peut facilement maximiser ce temps minimal pour obtenir la réponse présentée dans le cas c. Ce mécanisme s'étend sans difficultés aux faces et serait bien adapté aux mobiles utilisant une mécanique rigide. Une des conséquences est que la recherche de collisions devient un problème de minimisation globale, donc beaucoup plus coûteux. En effet, le déplacement d'un sommet en cas de collision dépend du temps trouvé pour la collision des arêtes incidentes. Si ce temps est calculé pour chaque arête (ou chaque face) en fonction de ses sommets, alors le mécanisme de minimisation doit se propager à l'ensemble du maillage. En cas de zone de collision large, le coût risque de devenir prohibitif, par contre la réponse calculée est bien plus précise.

Le second moyen d'éviter les disysmétries et de subdiviser le maillage des mobiles. Le système de suivi de particule supporte naturellement l'ajour (ou la supression) de particules au cours de la simulation. En ajoutant une particule au mileu de l'arête, on peut permettre sa déformation et obtenir une zone de contact plus précise, comme dans le cas d. En appliquant ce principe au deux sommets de l'arête, on obtient la réponse du cas e. Pour éviter l'ajout d'un trop grand nombre

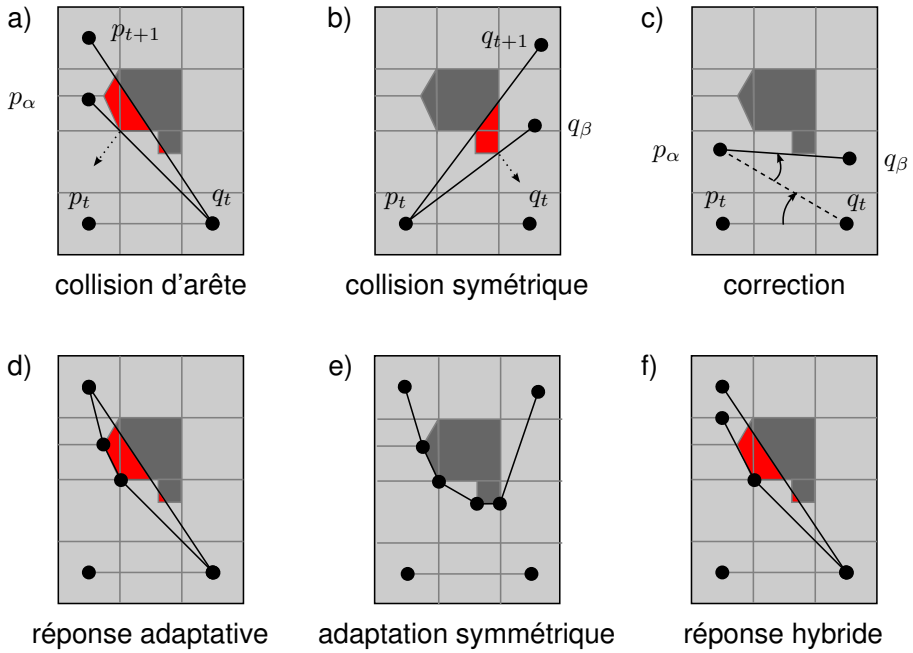


FIGURE 3.15 – Réponses possibles en cas de collisions d'arêtes. Les points p_α et q_β indiquent les mouvements calculés pour p_t et q_t par la recherche dichotomique.

de particules, une réponse hybride est possible. Elle consiste à ajouter une particule lorsque les deux arêtes générées sont suffisament grandes. Cette particule est ajouté au premier point de collision trouvé. Le sommet p_t et ensuite à nouveau déplacé vers p_{t+1} avec la réponse standard, comme montré dans le cas e.

Il faut noter ici que les particules ajoutées peuvent être ignorées par le système mécanique. Elles permettent juste de maximiser les déplacements autorisés en ajustant au mieux les zones de contacts aux obstacles. Les forces calculées sur les particules ajoutées peuvent par contre être ajoutées au système si ce dernier le supporte. Enfin si le système de simulation mécanique est multirésolution et permet la subdivision des cellules, alors les déformations du bord du mobile peuvent être complètement prises en compte.

3.6 Conclusion

Nous avons présenté dans ce chapitre, un mécanisme de suivi de particules, puis de maillages, dans un environnement subdivisé en cellules convexes. Ce mécanisme très général, nous a permis de développer des algorithmes de détection de collisions performants et adaptés aux simulations médicales où des environnements fortement déformables sont employés. Le système supporte les changements topologiques et détecte les collisions avec des temps de calcul indépendants, en pratique, de la complexité de la scène. Des réponses variées et adaptées à différents modèles physiques ont été proposées.

De nombreuses choses restent à développer. Parmi celles-ci, la détection des auto-collisions et des collisions entre objets multiples est incontournable. Ce sont des problématiques toujours très étudiées. L'approche présentée ici est compatible avec les méthodes historiques mentionnées en introduction. Les particules lors de leur suivi peuvent être liées aux cellules qui les contiennent. Ainsi chaque cellule (ou simplement chaque volume) aurait connaissance des particules qu'elle (ou il) contient. Il est alors aisé de savoir si des mobiles distincts coexistent dans ce volume. A

ce stade, des méthodes exactes, comme GJK ou Lin-Canny, peuvent être utilisées pour détecter les collisions entre ces mobiles, en partant des sommets correspondant aux particules.

Avec cette approche notre système remplacerait simplement unea hiérarchie de boites englobantes. La difficulté vient alors du nombre de particules présentes dans une cellule. Si celui-ci devient trop important, alors la comparaison des couples de mobiles distincts, augmentant de manière quadratique, deviendrait trop coûteuse. Il faudrait envisager de découper les cellules lorsqu'elles contiennent trop de particules. Cela conduirait à un suivi de particules multirésolution dont les applications sont d'ors et déjà prometteuses et inscrites dans le projet de recherche présenté à la fin de ce mémoire.

Chapitre 4

Reconstruction

Les travaux présentés dans ce chapitre font suite à la thèse de Dobrina Boltcheva soutenue en 2007 sous la direction de Dominique Bechmann et Sylvain Thery [16, 15]. Cette thèse proposait une méthode de maillage de la surface d'objets issus d'images médicales utilisant des diagrammes de Voronoï discrets. Elle présentait une méthode de reconstruction simple et robuste basée sur la croissance concurrente de ces régions de Voronoï. Les diagrammes obtenus permettaient au final de générer des triangulations de Delaunay sur le bord d'ensembles de voxels connexes.

A la suite de cette thèse, ces travaux en reconstruction se sont poursuivis dans le cadre de notre participation à deux projets : le projet ANR VORTISS (2006-2009) et le projet européen PASSPORT (2007-2010). Pour ces deux projets, notre participation était centrée sur la reconstruction d'organes du corps humain à partir de données médicales. Le projet VORTISS (Virtual Organ for Real Time Interactive Surgical Simulations) s'est déroulé en collaboration avec l'IRCAD à Strasbourg, l'équipe SIC du laboratoire XLIM à Poitiers et l'équipe projet INRIA Alcove à Lille. L'équipe SIC et l'EPI Alcove s'attachaient à définir des modèles physiques d'organes virtuels. Les données médicales étaient fournies par l'IRCAD qui assurait leur segmentation et l'étiquetage des structures anatomiques. Le projet STREP PASSPORT, supporté par l'IRCAD, avait pour but de proposer différents modèles biomécaniques du foie allant de l'échelle cellulaire à l'échelle humaine, ainsi que des méthodes d'acquisition de ces modèles utilisant différentes modalités d'acquisition.

Dans un premier temps, nous avons étendu les travaux de Dobrina Boltcheva au cas de la reconstruction surfacique simultanée de plusieurs organes, imbriqués les uns dans les autres, tout en exploitant plus à fond les modèles topologiques. Dans un second temps, la méthode a été généralisée à la construction de maillages volumiques à partir d'images voxel multi labels.

Ces travaux ont été implantés par Cyril Kern, ingénieur recruté en CDD sur ces deux projets et que j'ai encadré avec D. Bechmann et S. Thery. En parallèle, nous avons proposé des algorithmes de reconstruction adaptés aux réseaux vasculaires. Cette partie des travaux a été réalisée en partie par Younis Hijazi post-doctorant financé sur PASSPORT et encadré par D. Bechmann, S. Thery et moi-même.

Les travaux décrits dans ce chapitre ont été publiés dans une revue nationale [17] et une conférence internationale [69]. La section 4.1 présente le cadre d'application. La section 4.2 introduit l'outil des micro-cartes que nous utilisons dans ce cadre. Les sections 4.3 et 4.4 présentent les algorithmes de reconstruction surfacique et volumique proprement dits. Notez que les travaux concernant les maillages volumiques doivent encore être publiés. Pour les travaux sur les vaisseaux, le lecteur est renvoyé à l'article présenté en annexe B.

4.1 Introduction

Les travaux présentés dans ce chapitre visent à extraire un modèle géométrique maillé à partir d'images médicales 3D. Pour cela, une étape de *segmentation* est nécessaire. Pendant ce traitement, un label est associé à chaque *voxel* (pixel volumique) de l'image. De cette manière, les voxels ayant un label commun représentent une structure anatomique bien identifiée. Sa géométrie peut être simple, comme pour les organes ou les tumeur, ou plus complexe, comme pour les poumons, le cerveau ou les réseaux vasculaires. Par nature ces structures sont collées entre-elles, voire imbriquées les unes dans les autres. Les zones de contacts sont composées de 3 types d'intersections ou jonctions qui sont souvent décrites informellement : les 2-jonctions sont des morceaux de surfaces situées entre deux régions en contact ; les 1-jonctions sont des courbes situées à l'intersection de trois régions ou plus ; les 0-jonctions enfin correspondent aux points de rencontre de plusieurs 1-jonctions.

Nos travaux se démarquent des approches précédemment citées par deux aspects. D'une part, nous souhaitons générer des maillages dont la géométrie soit aussi proche que possible du bord des structures anatomiques segmentées. En particulier, les 1-jonctions devraient être discrétisées sous forme de lignes brisées, relativement *lisses*, dont les arêtes doivent appartenir aux bords des maillages volumiques ou surfaciques finaux. La topologie du résultat doit être garantie autour des ces jonctions et globalement.

D'autre part, notre cadre applicatif nous impose deux contraintes supplémentaires. Les données initiales sont souvent bruitées. Ainsi le résultat de la segmentation contient certainement des artéfacts. Ceux-ci, s'ils sont suffisamment petits, doivent disparaître naturellement durant la construction du maillage et surtout ne pas perturber les algorithmes. De plus, la procédure de reconstruction fait partie d'une chaîne complète de traitement partant des images du patient et allant jusqu'à leur exploitation par un médecin. L'ensemble des opérations – acquisition, segmentation, reconstruction et simulation – doit se faire dans des temps raisonnables, de l'ordre de la minute, et donc la production des maillages doit pouvoir être réalisée rapidement.

Pour certifier que les maillages générés définissent une topologie cohérente et soient compatibles entre eux, nous changeons la dimension du problème et construisons une partition volumique de l'image discrète sous forme d'une 3-variété combinatoire, même lorsqu'il s'agit de générer un maillage surfacique. Pour cela, nous considérons une région correspondant au fond de l'image et éventuellement aux zones de vide entre organes s'il y en a. Cette région peut être composée de plusieurs composantes connexes qui sont maillés en même temps et de la même manière que les autres régions.

Ainsi le maillage que nous générons est modélisé dans tous les cas par une carte combinatoire

de dimension 3. Si des maillages surfaciques sont souhaités pour l'application visée, alors chaque région sera modélisée par exactement un volume de cette 3-carte qui pourra être extrait facilement sous forme d'une carte combinatoire fermée de dimension 2. L'intérêt, encore une fois, est que toutes ces 2-cartes sont compatibles entre-elles le long des jonctions. Si un maillage volumique est souhaité, alors chaque région sera modélisée par un ensemble de tétraèdres connexes identifiés par le label de la région et cousus par ϕ_3. De plus, si on l'extrait, le bord extérieur de cet ensemble de tétraèdre correspond au maillage surfacique de la région.

Pour garantir la cohérence topologique durant toutes les étapes du traitement, nous avons choisi une approche consistant à équiper l'image d'une structure de 3-carte, dès le départ. Nous suivons en cela une approche similaire à celles utilisées dans [9, 61, 38]. Ces travaux introduisent des opérateurs de fusions et de découpes de cellules qui permettent de définir une cartographie hiérarchisée d'image 2D ou 3D. Ces pyramides de cartes ou de G-cartes sont utilisées pour définir des algorithmes robustes de segmentation [89, 5] et un modeleur discret [39].

A partir de cette topologie fine, nous proposons deux méthodes de reconstruction, l'une surfacique, l'autre volumique. Nous les détaillons ci-dessous après avoir présenté notre modèle de micro-carte.

4.2 Micro-carte et topologie discrète

Comme montré dans [55], une image peut être équipée d'une topologie discrète. Chaque pixel, en dimension 2, ou voxel, en dimension 3, défini un élément de surface ou de volume adjacent à 4 ou 8 voisins en dimension 2 et 6, 18 ou 26 voisins en dimension 3. Ces éléments ou cellules élémentaires sont représentés par des faces composées de 4 arêtes en dimension 2 ou par des volumes composés de 6 faces rectangulaires en dimension 3. Lors de la phase de segmentation ces cellules sont fusionnées pour former des régions connexes partageant un même label. Le résultat est une partition de l'image correspondant à ces régions.

Dans les travaux sur les pyramides de cartes ou de G-cartes cités plus haut, un des principaux buts est de définir des opérateurs de fusion et simplification de cellules permettant de définir différents niveaux de segmentation sur une image et offrant des garanties concernant la cohésion topologique de ces différents niveaux. Différentes représentations sont présentées (explicite, hiérarchique et implicite), ainsi que les algorithmes permettant de passer de l'une à l'autre.

Le modèle des micro-cartes présenté ici est une représentation simple et peu coûteuse en terme de consommation mémoire de la topologie d'une image segmentée formant une partition du plan ou de l'espace. Celle-ci doit nous permette d'exécuter efficacement les algorithmes de reconstruction présentés plus loin. En cela les micro-cartes sont une simplification de la représentation implicite des pyramides de cartes, ne possédant qu'un niveau hiérarchique. En particulier, nous ne définissons pas d'opérateurs de fusion ou de simplification aussi élaborés que ceux cités plus haut.

4.2.1 Micro-carte de dimension 2

Informellement, une micro-2-carte est une 2-carte plaquée sur une image dont les faces, arêtes et sommets correspondent aux pixels de l'image. Au départ, une micro-carte possède une face par pixel. La seule opération autorisée est la suppression d'une arête. Cette arête peut-être placée entre deux faces ce qui provoque leur fusion ou n'appartenir qu'à une face ce qui est une forme de simplification. Les images que nous voulons manipuler étant relativement grande, pour réduire les coûts mémoire, la carte n'est pas explicitement stockée. Seule l'information de suppression d'arête l'est. En pratique, nous stockons un booléen par brin indiquant s'il s'agit d'un brin supprimé ou non. Les informations topologiques sont recalculées à la volée.

Formellement, un *micro-brin* est un triplet (x, y, i) où x et y sont des entiers représentant les coordonnées d'un pixel et i est un entier compris entre 0 et 3 indiquant la position du brin dans

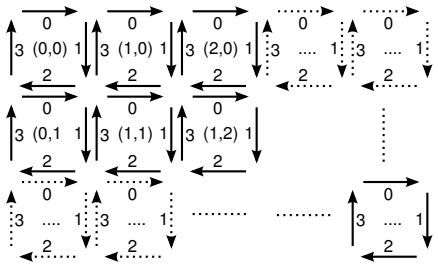


FIGURE 4.1 – Les micro-brins d'une micro-carte complète

la face (voir figure 4.1). Deux permutations ϕ'_1 et ϕ'_2 sont définies sur les micro-brins telles que :

$$\begin{aligned}
\phi'_1((x,y,i)) &= (x,y,(i+1)\%4)\\
\phi'_2((x,y,0)) &= (x,y-1,2)\\
\phi'_2((x,y,1)) &= (x+1,y,3)\\
\phi'_2((x,y,2)) &= (x,y+1,0)\\
\phi'_2((x,y,3)) &= (x-1,y,1)
\end{aligned}$$

Clairement ϕ'_2 est une involution, c'est-à-dire que $\phi'_2(\phi'_2(b)) = b$. Tout aussi trivialement, ϕ'_1 est une permutation dont tous les orbites sont composés de 4 micro-brins de même coordonnées et de positions successives 0, 1, 2 et 3. Sans limitation sur les valeurs de x et y les micro-brins définissent implicitement une grille régulière infinie.

Une micro-2-carte complète $M_{X,Y}$ sur une image de taille $X \times Y$ pixels est une 2-carte (B', ϕ'_1, ϕ_2) définie par l'ensemble des micro-brins $B' = \{(x,y,i)\}$ tels que $i \in \{0,\ldots,3\}$, $x \in \{0,\ldots,X-1\}$ et $y \in \{0,\ldots,Y-1\}$, par la permutation ϕ'_1 et par l'involution ϕ_2 restreignant x et y aux valeurs permises en ajoutant des points fixes à ϕ'_2 de la manière suivante :

$$\begin{aligned}
\phi_2((x,y,0)) &= \text{si } (y-1) \geq 0 \text{ alors } \phi'_2((x,y,0)) \text{ sinon } (x,y,0)\\
\phi_2((x,y,1)) &= \text{si } (x+1) < X \text{ alors } \phi'_2((x,y,1)) \text{ sinon } (x,y,1)\\
\phi_2((x,y,2)) &= \text{si } (y+1) < Y \text{ alors } \phi'_2((x,y,2)) \text{ sinon } (x,y,2)\\
\phi_2((x,y,3)) &= \text{si } (x-1) \geq 0 \text{ alors } \phi'_2((x,y,3)) \text{ sinon } (x,y,3)
\end{aligned}$$

Clairement la micro-carte complète $M_{X,Y}$ est une 2-carte possédant XY faces de 4 micro-brins.

Une micro-2-carte $M_{X,Y,existe}$ est une carte complète $M_{X,Y}$ dans laquelle certains brins sont marqués comme étant supprimés. Le prédicat $existe(b)$ indique si un brin existe ou a été supprimé dans la carte. Soit B' l'ensemble des micro-brins de $M_{X,Y}$, notons $B = \{b \in B', existe(b)\}$ l'ensemble des brins non supprimés. La micro-carte $M_{X,Y,existe}$ définit une 2-carte (B, ϕ_1, ϕ_2) où l'involution ϕ_2 est la restriction à B de celle définit précédemment et la permutation ϕ_1 est définie sur B récursivement de la manière suivante :

$$\phi_1(b) = \text{si } existe(\phi'_1(b)) \text{ alors } \phi'_1(b) \text{ sinon } \phi_1(\phi_2(\phi'_1(b)))$$

Cette nouvelle permutation prend en compte les brins marqués comme supprimés en *sautant* par dessus eux (voir figure 4.1). Cela correspond au fait de tourner autour du sommet jusqu'à trouver une arête existante. On notera que cette formulation, choisie ici pour sa simplicité, ne termine que si le brin de départ existe, c'est-à-dire appartient à B.

Pour que la 2-carte décrive une partition totale de l'image, elle ne doit pas contenir de *trous*, c'est-à-dire de pixels n'appartenant à aucune face. Formellement, le bord de la carte doit

comporter une unique face plongée sur le bord de l'image. Pour cela elle doit vérifier deux contraintes d'intégrité simple. D'une part, les brins du bord de l'image, c'est-à-dire ceux pour lesquels $x = 0$, $x = X - 1$, $y = 0$ ou $y = Y - 1$, ne peuvent être supprimés. D'autre part, les seuls points fixes par ϕ_2 sont ces brins du bord. Cela signifie concrètement que deux brins d'une micro-arête $\{b, \phi_2(b)\}$ doivent toujours être supprimés simultanément.

4.2.2 Micro-carte de dimension 3

De la même manière qu'en dimension 2, une micro-3-carte est une 3-carte plaquée sur une image volumique telle que chacun de ses volumes corresponde à un voxel de l'image. Les faces des volumes correspondent aux *surfels* ou éléments de surfaces entre 2 voxels 6-adjacents.

Un micro-brin de dimension 3 est un quadruplet (x, y, z, i) où x, y et z sont les coordonnées du voxel dans l'image et où i est un entier compris entre 0 et 23 indiquant la position du micro-brin dans le voxel. Trois permutations ϕ'_1, ϕ'_2 et ϕ'_3 sont définies pour passer d'un micro-brin à l'autre. ϕ'_1 et ϕ'_2 sont simplement des permutations sur la valeur de la position i qui peuvent être lues directement sur le schéma. Pour ϕ'_3, il s'agit de passer au voxel voisin en incrémentant ou décrémentant l'une des coordonnées et en prenant pour i la position symétrique dans le voxel.

Une micro-3-carte complète $M_{X,Y,Z}$ sur une image 3D de taille $X \times Y \times Z$ est une 3-carte définie par $(B', \phi'_1, \phi'_2, \phi_3)$. Ici encore B' est l'ensemble des micro-brins ayant des valeurs de coordonnées correctes et ϕ_3 est définie à partir de ϕ'_3 en ajoutant des points fixes au bord de l'image.

Une micro-3-carte $M_{X,Y,Z,existe}$ est définie de la même manière qu'en dimension 2, en considérant un prédicat d'existence des brins. Notons ici que pour assurer la validité du modèle et donc de la partition volumique de l'image, les micro-brins sont supprimés face par face. Ainsi, deux faces d'orientation opposées comprises entre 2 voxels 6-voisins doivent être supprimées simultanément. Cela correspond à la suppression de 8 micro-brins cousus par ϕ_1 et ϕ_3.

La permutation ϕ_2 est définie à partir de ϕ'_2 pour *sauter* les faces supprimés, en combinant ϕ'_2 et ϕ'_3. Cela revient à tourner autour des arêtes pour trouver une face non supprimée. La permutation ϕ_1 est simplement la restriction de ϕ'_1 aux brins existants.

De manière plus intuitive, une micro-3-carte décrit une partition en volumes d'une image 3D. Lorsqu'une face est comprise entre 2 volumes distincts, sa suppression provoque la fusion de ces 2 volumes. Sinon, elle correspond à la simplification de l'intérieur d'un volume. Au final, dans une micro-3-carte les volumes sont séparés par des éléments de surfaces composés de surfels collés les uns aux autres par ϕ_2. Les faces d'une micro-3-carte à ce stade ne peuvent être fusionnées entre-elles.

4.2.3 Implantation

Le but des micro-cartes est d'encoder la topologie d'une partition d'une image. Dans de nombreuses applications, la taille des images dépasse 1024^3 voxels ce qui rend l'usage de structures topologiques classiques impossible en pratique. Ainsi une attention particulière est apportée à l'implantation des micro-cartes, pour qu'elles puissent être utilisées efficacement sur des données réelles.

D'un point de vue pratique, les micro-brins peuvent être représentés par des entiers. Ainsi, en dimension 2, le brin (x, y, i) peut être encodé par l'entier $b = (y * X + x) * 4 + i$ compris entre 0 et $4XY - 1$. Les entiers x, y et i peuvent être extrait par des combinaisons adéquates de divisions et de masques logiques. En dimension 3, on fait intervenir z et la profondeur de l'image, de la même manière. Les fonctions ϕ_i correspondent à de simples calculs arithmétiques. Elles sont traduites pour être exécutées directement sur la représentation entière ou définies plus simplement sur les triplets. L'usage de l'une ou l'autre version dépend des algorithmes et de leurs besoins en termes d'accès mémoire.

Le prédicat *existe* peut simplement être stocké dans un tableau de booléens. Cependant lorsque des algorithmes géométriques doivent être exécutés sur ce type de structures, ce qui est le cas dans nos travaux, il faut être capable d'associer des informations de plongement (coordonnées, courbures, labels, etc.) aux micro-brins. En dimension 2, cela ne pose pas de problème, il suffit d'allouer pour chaque plongement un tableau de taille $4XY$. Chaque micro-brin, supprimé ou non, aura une entrée dans ce tableau.

En dimension 3 par contre, la taille des données interdit une implantation aussi simple et seuls les plongements des micro-brins existants doivent être alloués et stockés. C'est le principal but de l'introduction des micro-cartes. Les informations de plongement peuvent être portées par les voxels, les sommets de la carte, les faces ou les arêtes. En pratique, les plus gros besoins concernent les voxels.

Pour le plongement des voxels, les brins sont regroupés par 24 (les 24 brins ayant les mêmes coordonnées et des positions différentes). Le tableau stockant le prédicat *existe* est remplacé par un tableau de pointeur vers le plongement de voxel. Si les 24 brins sont supprimés, ce qui est vrai dans la majorité des cas, ce pointeur est laissé à la valeur nulle. Sinon, le plongement est alloué et contient les 24 booléens des micro-brins concernés. Lorsque les espaces vides sont importants, ce tableau peut-être remplacé par un octree dont les feuilles contiennent les plongements des brins des voxels existants.

Les autres plongements nécessitent des accès moins fréquents et peuvent donc être stockés dans une table associative utilisant l'entier codant le micro-brin comme clé. Si les besoins de l'application visée le nécessitent, la même technique que pour les plongements de voxels peut-être utilisée. Nous n'irons pas plus loin concernant les détails d'implantation. Le but était simplement de montrer que l'on est capable de gérer et stocker des plongements variés et complets pour des images volumiques de très grande taille.

Pour terminer, considérons les temps d'accès aux informations topologiques. On pourrait considérer que la simplicité des opérations de simplifications autorisées limite trop notre modèle. Cela serait le cas si l'on souhaitait en faire un outil de modélisation ou d'analyse multirésolution. Les travaux sur les pyramides de cartes déjà mentionnés, on d'ailleurs proposés des outils très puissant pour cela. L'intérêt principal de notre approche est que les parcours topologiques restent localement exécutables en temps constant. Précisément, les relations topologiques devant être calculées à la volée, c'est-à-dire ϕ_1 en dimension 2 et ϕ_2 en dimension 3, nécessitent dans le pire des cas de parcourir 4 micro-brins. Cela nous permet de parcourir les surfaces de contact entre les régions de l'image de manière optimale et garantit donc la performance, en terme de temps de calcul, des algorithmes de reconstruction décrits dans les sections suivantes.

Les micro-cartes de dimension 2 ont été présentées par souci de pédagogique. Dans la suite seule des images 3D et donc des micro-cartes de dimension 3 sont considérées. Certaines figures seront tout de même présentées en dimension 2 quand cela permet rendre les schémas plus lisibles. Nous appelons *micro-modèle* les objets géométriques dont la topologie est décrite par une micro-carte. Pour éviter les ambiguïtés, nous appelons micro-sommet et micro-arête, les sommets et arêtes du micro-modèle (ou de la micro-carte) pour les distinguer des sommets et arêtes du maillage généré.

4.3 Reconstruction surfacique

Nous présentons dans cette section un algorithme de reconstruction permettant de mailler la surface de régions issues d'une segmentation d'image 3D. Dans cette version l'intérieur des régions n'est pas maillé, mais le résultat est bien une 3-carte pour assurer la compatibilité des maillages au niveau des jonctions.

4.3.1 Extraction du micro-modèle depuis une image

La première étape de la reconstruction consiste à extraire le micro-modèle de l'image segmentée. Au départ chaque pixel ou voxel s'est vu attribué un label correspondant à une structure anatomique. Si on considère une micro-carte complète plaquée sur l'image, alors l'extraction consiste simplement à supprimer les micro-brins situés entre deux pixels (en 2D) ou voxels (en 3D) de même label. Cela revient à fusionner les faces ou volumes appartenant à une même structure. Notons qu'avec cette approche seule la 4-connectivité en 2D et la 6-connectivité en 3D sont considérées, les autres connectivités n'ayant d'ailleurs pas de sens pour des images multi-labels.

D'un point de vue pratique, notre micro-modèle est construit dans l'autre sens, pour éviter des traitements inutiles. Nous partons d'une micro-carte où tous les brins sont marqués comme étant *non existants*. Ensuite, un simple parcours de l'image permet de trouver les brins situés à la frontière des structures anatomiques, c'est-à-dire entre deux pixels ou voxels de labels différents. Ces brins sont marqués comme *existants* et leurs plongements sont alloués à la volée.

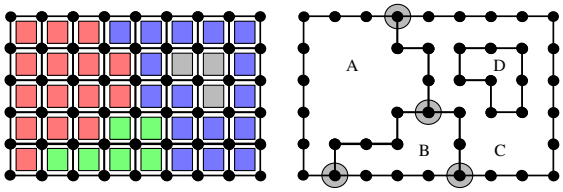


FIGURE 4.2 – Un micro-modèle 2D : (à gauche) l'image de départ ; (à droite) les frontières des 4 objets présents. Les sommets soulignés correspondent aux 0-jonctions. Ils séparent le bord en 7 courbes correspondant aux 1-jonctions.

La gestion des inclusions, comme dans le cas des région C et D de la figure 4.2, dépend des applications. Les bords de ces régions peuvent simplement être maillés séparément, l'information d'inclusion étant calculée par ailleurs. C'est le comportement par défaut de notre algorithme. En cas de besoin, des éléments de maillage reliant les différentes composantes connexes peuvent être créés. Pour cela, une courbe (ou une surface en 3D) peut être *dessinée* sur l'image en marquant comme existant certains brins qui aurait dû être supprimés. Les ponts ainsi créés sont maillés en même temps que les bords par notre algorithme sans traitement supplémentaire.

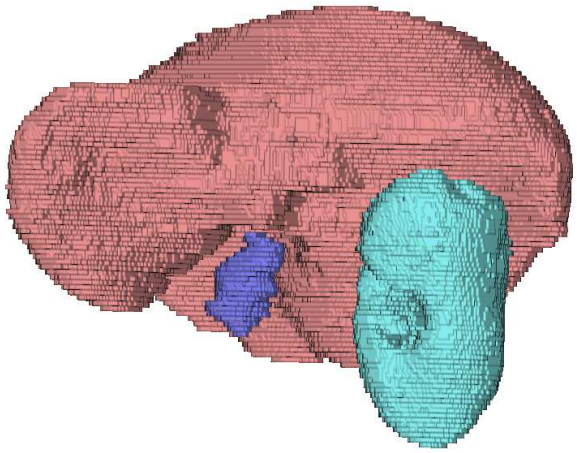


FIGURE 4.3 – Un micro-modèle montrant un foie en rouge, un rein droit en bleu clair et une vésicule biliaire en violet. L'intérieur des objets est creux.

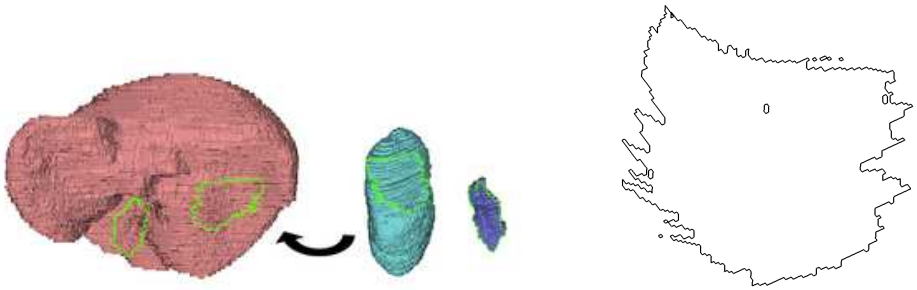


FIGURE 4.4 – Les 1-jonctions entre structures anatomiques (en vert) et une vue détaillée et mise à plat des 1-jonctions entre le foie et le rein (à droite).

En dimension 3, les zones du micro-modèle à mailler correspondent aux 2-jonctions. Comme on le voit dans la figure 4.3, ces zones forment des morceaux de surfaces séparées par des courbes discrètes. Les 1-jonctions, représentées en vert figure 4.4, sont les zones de contact entre plus de deux objets. Elles forment un ensemble de courbes séparant les 2-jonctions. Leur topologie est celle d'un 1-complexe cellulaire que nous appelons le *complexe séparateur*. Celui-ci est composé de plusieurs cycles de micro-brins. Les plus petits cycles comme dans l'exemple de la figure 4.4 sont clairement dus au bruit et disparaissent naturellement par la suite.

4.3.2 Principe général de la méthode

Notre but est de mailler les surfaces correspondant aux 2-jonctions, de manière à ce que leurs bords coïncident le long du complexe séparateur, comme cela est montré dans la figure 4.5. L'idée est de dessiner sur ces surfaces un diagramme de Voronoï discret, avant d'en obtenir une triangulation de Delaunay. Les sommets de cette triangulation seront les centres des régions de Voronoï construites.

Rappelons que le diagramme de Voronoï est un partitionnement en régions tel que la région associée à un sommet, contient les points plus proches de ce sommet que de tout autre. Le calcul du diagramme de Voronoï peut être effectué de façon exacte par la géométrie algorithmique [7], ou de façon approchée avec une image ou carte de distance [18]. C'est cette dernière approche que nous exploitons ici.

FIGURE 4.5 – Trois étapes du maillage de deux objets adjacents.

Pour obtenir ces régions, nous plaçons sur les surfaces à mailler un ensemble de *graines* autour desquelles nous faisons croître des disques topologiques. Ces croissances parallèles se poursuivent jusqu'à ce que l'ensemble du micro-modèle soit recouvert. Par construction, les sommets d'une région sont plus proches de la graine dont leur région est issue que des graines des régions voisines. A la fin, une arête de Delaunay est crée entre chaque couple de graines dont les régions sont adjacentes, les graines formant alors les sommets de cette triangulation.

Notons ici que le micro-modèle initial est déjà une 3-carte combinatoire. Les algorithmes qui suivent sont, pour l'essentiel, basés sur des parcours de morceaux de surfaces. Ceux-ci s'expriment

donc comme le suivi des relations topologiques ϕ_1 et ϕ_2, la relation ϕ_3 permettant de passer au volume adjacent. Les méthodes travaillant directement sur les données discrètes doivent en permanence, lors de tels parcours, considérer les labels des voxels traversés et leur connexité. Cela les rend peu lisibles et difficiles à contrôler. Notre approche garantit la topologie de manière globale et bénéficie dès le départ de l'efficacité des parcours dans les modèles ordonnés.

4.3.3 Qualité géométrique de la discrétisation

La sélection des graines est une étape cruciale de l'algorithme. Leurs positions et leur distribution influencent grandement la qualité des approximations géométriques obtenues. Pour obtenir une triangulation régulière des objets, nous fixons, entre deux graines, une distance minimale qui assure leur distribution uniforme sur la surface du micro-modèle. Ainsi lorsqu'une graine est placée sur un micro-sommet s, nous interdisons le placement d'une autre graine dans un voisinage de rayon r autour de s, en marquant les micro-brins concernés. Ce rayon impose une distance minimale entre deux graines et donc la longueur minimale des arêtes du résultat.

Pour évaluer ces rayons, une distance euclidienne n'est pas adaptée ici. Nous utilisons une *distance géodésique anisotrope* définie entre deux micro-sommets comme étant la longueur du plus petit chemin de micro-arêtes pondérées par les facteurs d'anisotropie de l'image (la taille des voxels). Cette distance peut être calculée sur une courbe discrète, en suivant les 1-jonctions, ou sur la surface du micro-modèle. Elle permet de conserver des détails fins en n'interdisant pas de placer des graines sur des micro-sommets proches, en terme de distance euclidienne, mais placés sur des replis différents de la surface. En pratique, cette distance n'est jamais calculée point à point car cela nécessiterait la recherche d'un chemin minimal ce qui est relativement coûteux. Elle est évaluée de manière incrémentale lors de la traversée des éléments de surface qui sont implantés par des parcours en largueur autour des micro-sommets.

L'ordre utilisé pour placer les graines est également important. Une distribution non ordonnée ou aléatoire des graines risque de gommer des détails de la surface et de lisser les bords. Pour éviter cela, les graines sont placées en priorité aux endroits de forte courbure ce qui assure la présence des détails correspondants dans le résultat. La méthode pour calculer la courbure est indifférente. Notons cependant qu'il s'agit ici d'une courbure discrète et que son évaluation doit prendre en compte un voisinage du sommet dont la taille peut éventuellement être liée à r.

4.3.4 Qualité et cohérence de la topologie

A ces considérations géométriques s'ajoute le fait que les maillages obtenus doivent coïncider au niveau du complexe séparateur. Cela introduit plusieurs contraintes dans la triangulation de Delaunay en construction. D'une part, les courbes discrètes formant les 1-jonctions doivent être maillées par des lignes polygonales continues. D'autre part, les maillages des éléments de surface doivent s'appuyer de manière cohérente sur ces lignes polygonales. Pour garantir cela, trois règles simples, concernant le placement des graines et la croissance des régions, sont nécessaires.

Il faut premièrement s'assurer que deux graines consécutives placées sur une 1-jonction produisent toujours une arête de Delaunay. Cela nécessite qu'elles appartiennent à deux régions de Voronoï adjacentes. Il suffit pour cela d'interdire qu'une région de Voronoï traverse une 1-jonction, sauf quand elle est elle-même issue d'une graine placé sur cette 1-jonction.

Deuxièmement, les 1-jonctions forment des courbes qui se rencontrent en des points singuliers ou 0-jonctions. Les lignes polygonales qui les discrétisent doivent faire de même. Une graine est systématiquement placée sur chaque 0-jonction ce qui impose la présence d'un sommet de Delaunay à cet endroit. Notons ici que cette contrainte topologique prend le pas sur les considérations géométriques décrites plus haut. Ainsi deux graines peuvent être placées à une distance inférieure au rayon r si les 0-jonctions les portant sont proches.

Troisièmement, une 1-jonction est formée de micro-brins adjacents à 3 volumes ou plus et donc adjacente à au moins 3 surfaces. A cet endroit, nous avons donc des arêtes non variétés.

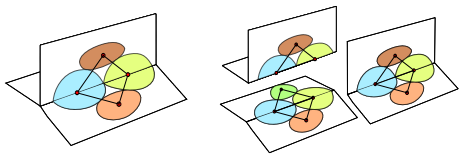


FIGURE 4.6 – Régions de Voronoï étendue le long du 1-complexe séparateur : (à gauche) les régions de Voronoï et la triangulation obtenue ; (à droite) les disques topologiques dans les trois volumes adjacents.

Ainsi le micro-maillage n'est localement pas une surface (ce qui est normal pour une 3-carte). La notion de région de Voronoï, définie auparavant comme un disque topologique autour d'une graine, doit être étendue de manière suivante. Une région de Voronoï autour d'une 1-jonction est l'union des disques de Voronoï définis sur la surface des volumes incidents à cette 1-jonction. Naturellement ces disques correspondent deux à deux autour de brins cousus par ϕ_3, comme illustré figure 4.6.

4.3.5 Approximation discrète des régions de Voronoï

La génération des régions de Voronoï utilise les principes suivants. Chaque nouvelle graine se voit attribuer une couleur distincte. Une région de Voronoï est coloriée sur la surface autour de la graine dont elle est issue. Les brins de volumes adjacents, cousus par ϕ_3 sont systématiquement coloriés en même temps, pour que les régions soient compatibles sur toutes les surfaces. Durant cette croissance la distance à la graine de départ est évaluée de manière incrémentale et stockée dans chaque micro-sommet parcouru. Lors de sa croissance, une région progresse sur les micro-sommets non coloriés et sur les sommets déjà coloriés dont la distance stockée est plus grande que la distance à la graine courante, en cours de calcul. Les régions se *grignotent* ainsi entre-elles ce qui leur assure des tailles homogènes. Cette croissance ne traverse jamais de 1-jonctions pour assurer que les régions forment bien des morceaux de surfaces.

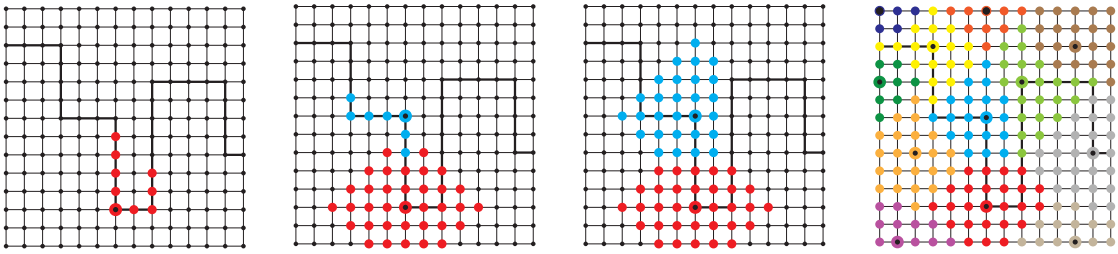


FIGURE 4.7 – Croissance de régions discrètes : la ligne en gras représente une portion du 1-complexe séparateur. De gauche à droite, le micro-modèle initial et la première croissance à partir d'une graine, le long de la 1-jonction ; la croissance de région à partir de cette graine et celle de la seconde grain ; la croissance de région à partir de cette graine, bloquée au niveau de la 1-jonction ; enfin, le résultat final de la croissance de toutes les graines.

Pour assurer le placement correct des graines et obtenir des régions de Voronoï vérifiant immédiatement les contraintes énoncées plus haut, nous utilisons l'algorithme suivant pour les placer. Dans un premier temps, une graine est placée sur chaque 0-jonction et des régions immédiatement coloriées autour de celles-ci. Dans une seconde phase, les micro-sommets des 1-jonctions sont triés par courbures décroissantes. Lorsqu'une graine est placée sur une 1-jonction, la croissance de région est d'abord effectuée sur la courbe correspondante. Ensuite la croissance s'opère sur les surfaces incidentes avec l'algorithme énoncé ci-dessus et donc sans repasser par cette 1-jonction.

Enfin, les graines sont placées sur les surfaces restantes, encore par courbures décroissantes.

L'algorithme est illustré en dimension 2 figure 4.7. Le résultat est une approximation d'un diagramme de Voronoï défini sur l'ensemble des surfaces des structures anatomiques modélisées. Les régions de Voronoï correspondent exactement sur les volumes adjacents. Ce diagramme est contraint au niveau des 1-jonctions de telle sorte que la triangulation de Delaunay de chaque surface s'appuie sur des arêtes communes portées par les 1-jonctions. Un exemple de diagramme obtenu est montré figure 4.8.

FIGURE 4.8 – Approximation discrète des régions de Voronoï sur le micro-modèle de la figure 4.3.

4.3.6 Extraction du graphe de Voronoï

L'étape suivante de la méthode de reconstruction est l'extraction du graphe de Voronoï, à partir du diagramme discret. Rappelons que les arêtes de Voronoï séparent deux régions adjacentes. Etant donnée la nature discrète des régions manipulées, ces arêtes sont ici des courbes discrètes. Les sommets du graphe de Voronoï correspondent aux lieux où les arêtes de Voronoï se touchent.

Le diagramme de Voronoï est formé de micro-faces dont les quatre micro-sommets portent les couleurs de leur région. Elles peuvent être classifiées ainsi : une micro-face dont les 4 sommets possèdent la même couleur appartient à l'intérieur d'une région du graphe de Voronoï ; une micro-face dont les 4 sommets ont exactement 2 couleurs correspond à une arête du graphe de Voronoï ; une micro-face dont les 4 sommets ont 3 ou 4 couleurs différentes correspond à un sommet du graphe de Voronoï.

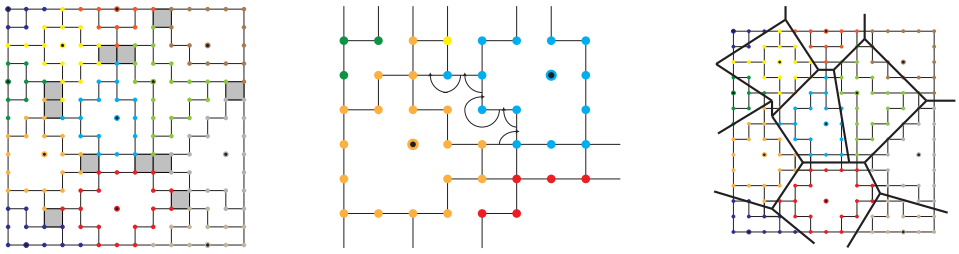


FIGURE 4.9 – Simplification du diagramme de Voronoï : le graphe de Voronoï discret, détails d'une arête de Voronoï et le graphe final.

Comme nous avons utilisé le prédicat *existe* sur les micro-brins des arêtes, pour définir les micro-cartes, nous définissons un prédicat $fusion$ sur les micro-arêtes séparant deux micro-faces ayant la même classification. Ainsi, $fusion(b)$ est vrai si les deux micro-faces adjacentes au brin b appartiennent à la même région ou arête du graphe de Voronoï. Avec ce prédicat, nous

définissons une nouvelle partition implicite sur les surfaces, représentée figure 4.9. Elle correspond à la simplification du diagramme de Voronoï discret et forme un graphe de Voronoï discret dont les arêtes sont des courbes discrètes. Un simple parcours de ce graphe permet d'extraire le graphe de Voronoï continu recherché.

4.3.7 Triangulation de Delaunay

La dernière étape de la reconstruction consiste à construire le maillage par dualité à partir du graphe de Voronoï. Ce modèle est une 3-variété topologique dans laquelle chaque volume correspond à un objet. Le bord de ces volumes qui sont des 2-variétés combinatoires, correspondent aux surfaces à mailler. Un exemple de cette construction est donné sur la figure 4.10, sur un petit bout de surface. Notons que les graines placées en priorité sur les 1-jonctions sont correctement connectés par des arêtes de la triangulation duale. Ces arêtes, dessinées en rouge, maillent le 1-complexe séparateur et assurent la qualité géométrique des interfaces entre objets.

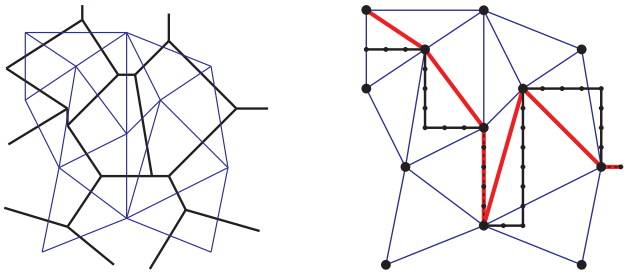


FIGURE 4.10 – Triangulation de Delaunay finale : à gauche, le dual du graphe de Voronoï et, à droite, le maillage de la 1-jonction.

Le dual géométrique du graphe de Voronoï est connu comme la *triangulation de Delaunay*, mais ce dual est défini usuellement sur une surface. Le passage au dual dans la 3-carte ne donnerait absolument pas le résultat escompté. En fait, le graphe volumique obtenu précédemment est constitué d'un ensemble de graphes surfaciques collés entre eux au niveau des 2-jonctions qui ont des sommets, arêtes et faces communs.

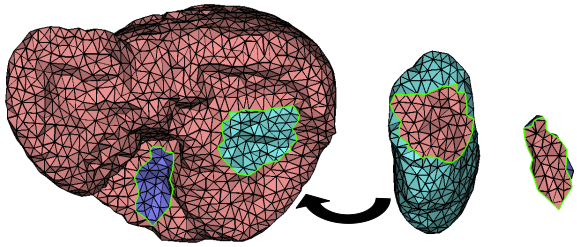


FIGURE 4.11 – Maillage obtenu à partir du micro-modèle de la figure 4.4 avec un rayon de résolution $r = 15$. Le 1-complexe séparateur est surligné en vert.

Pour obtenir le maillage de Delaunay, nous prenons le graphe surfacique dual dans chaque objet. Une dernière étape recolle ces morceaux pour obtenir le maillage volumique final. En pratique le recollage est immédiat car les paires de cellules communes sont connues avant le passage au dual. Ce passage au dual dans chaque composante volumique fonctionne uniquement parce que nous nous sommes assuré que seules des régions, deux à deux adjacentes, étaient présentes au niveau des 1-jonctions. Ainsi dans chaque volume, les cellules traversant les 1-jonctions sont

des faces et des arêtes. Le passage au dual les transforment en sommets correctement reliés par des arêtes. La figure 4.11 montre le maillage volumique obtenu à partir du micro-modèle du foie, du rein et de la vésicule biliaire de la figure 4.3.

4.3.8 Expérimentation

L'algorithme de reconstruction multi-objets que nous avons présenté ici a été implanté sur la plateforme de modélisation à base topologique développée dans notre équipe. Nous avons testé cet algorithme sur différents types d'images médicales segmentées.

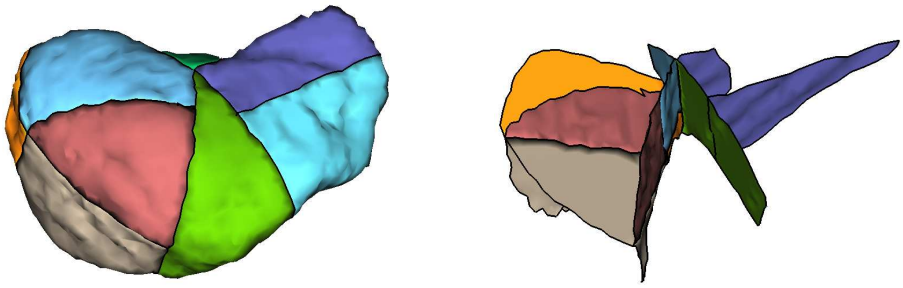


FIGURE 4.12 – Maillage des 8 segments anatomiques du foie et leurs interfaces internes.

La figure 4.12 montre, par exemple, le modèle reconstruit à partir d'un foie découpé en ses huit segments anatomiques et les interfaces entres ces segments. Un autre exemple est montré sur la figure 4.14(a) où plus de 20 structures anatomiques ont été reconstruites simultanément à partir d'une image segmentée de l'abdomen.

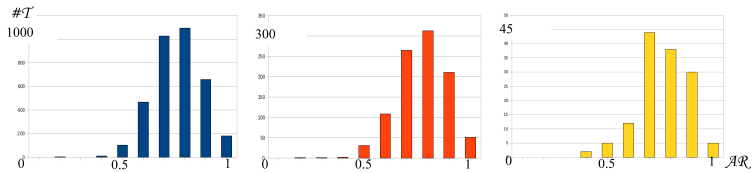


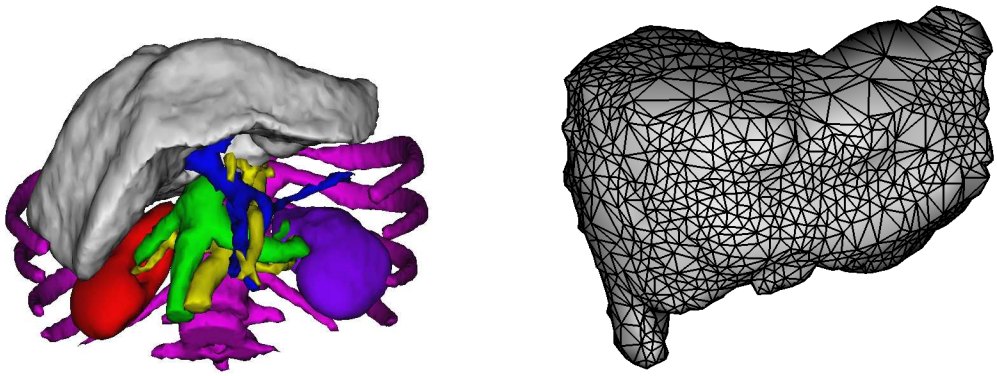
FIGURE 4.13 – Aspect ratio du foie, du rein et de la vésicule biliaire de la figure 4.11.

Pour évaluer la qualité géométrique des maillages obtenus, nous avons calculé l'aspect ratio σ des triangles. Pour un triangle τ, $\sigma(\tau)$ est le rapport entre sa plus petite et sa plus grande arête. Il est proche de 1 pour des triangles réguliers et vaut 1 pour les triangles équilatéraux. La figure 4.13 montre que la majorité des triangles des maillages du foie, du rein et de la vésicule biliaire de la figure 4.11 présente un aspect ratio supérieur à 0.5. Cela montre que notre méthode construit des maillages de très bonne qualité.

4.3.9 Adaptabilité

Nous avons présenté une version uniforme de la reconstruction. Le rayon discret des régions de Voronoï construites au départ est le même partout. Or ce rayon n'intervient dans l'algorithme que pour marquer les micro-brins qui ne peuvent plus être choisis comme sommet du maillage. Ce n'est donc pas un paramètre global de la méthode.

Une extension très simple consiste à utiliser un rayon différent pour chaque région. La valeur de ce rayon peut être donnée par la nature de l'objet auquel appartient le pointel ou par la courbure locale. Le premier choix permet d'obtenir des maillages plus ou moins fins, mais encore



(a) Maillage des structures anatomiques de l'abdomen.

(b) Maillage adaptatif du foie.

FIGURE 4.14 – Exemples de maillages.

compatibles, des différents organes. Par exemple un maillage grossier pour le foie et un maillage plus fins pour la trachée. Le second choix permet d'obtenir un maillage dont la taille des triangles s'adapte à la courbure locale des objets. Ces deux extensions peuvent bien sûr être combinées. Pour obtenir l'exemple représenté sur la figure 4.14(b), nous avons adapté la distribution des graines à la courbure locale.

4.4 Reconstruction volumique

La génération de maillages volumiques utilise une méthode similaire, plus simple par certains aspects. Elle consiste à construire un diagramme de Voronoï volumique discret contraint par les bords des structures anatomiques, c'est-à-dire tel qu'une région de Voronoï ne contienne que des voxels d'un même label. L'algorithme qui n'a pas encore été publié, même si son développement est achevé, est simplement esquissé dans la présente section.

L'algorithme commence directement sur le micro-modèle défini par l'image segmentée, sans fusionner les régions ayant le même label, c'est-à-dire en conservant tous les voxels internes aux régions à mailler. Les voxels extérieurs ou ceux qui n'appartiennent pas aux régions d'intérêt sont fusionnés pour former une grande région non maillée, éventuellement non connexe.

Les régions de Voronoï sont obtenues en faisant croître des sphères de rayon r autour des graines sélectionnées. Les zones de contact entre ces sphères discrètes forment par la suite le graphe de Voronoï volumique. Le choix des graines est similaire au cas surfacique. Nous commençons par placer des graines aux 0-jonctions. Puis le long des 1-jonctions, en s'assurant qu'elles sont à nouveau complètement couvertes par des régions adjacentes.

Viennent ensuite les 2-jonctions. Pour que le maillage dual corresponde à ce que l'on souhaite, nous ajoutons une contrainte à la croissance des régions volumiques. Celles-ci ne peuvent passer au travers de 2-jonctions. Une croissance surfacique est donc réalisée d'abord, pour interdire la présence de sommets de Voronoï au niveau des interfaces entre objets. Ces sommets se transformeraient en volumes dans le graphe dual ce qui ne serait pas compatible avec les frontières entre objets. Finalement, les intérieurs des volumes sont remplis de sphères discrètes qui achèvent le placement des graines.

Après la croissance de toutes les régions, nous obtenons une partition colorée des voxels de l ?image, compatible avec les toutes les jonctions. Les voxels sont également classifiés en fonction de la couleur de leurs micro-sommets. Si un voxel possède 8 sommets de la même couleur, alors il est à l'intérieur d'une région de Voronoï. Si ses sommets possèdent exactement deux couleurs,

alors le voxel appartient à une face du graphe de Voronoï. Les autres voxels appartiennent aux arêtes ou sommets de ce graphe discret. Les arêtes sont détectées car elles forment des suites de voxels partageant les mêmes couleurs, car incidentes aux mêmes faces.

Les arêtes du graphe de Voronoï discret sont des courbes discrètes. Les faces sont des surfaces discrètes d'épaisseur 1. Pour extraire, le graphe continu, il suffit de parcourir les bords de ces faces. Le nombre d'arêtes discrètes rencontrées permet de créer des polygones topologiques correspondants. L'ordre des faces autour des arêtes permet de coudre ces polygones ensemble autour de leurs arêtes. Au final, on obtient une 3-carte modélisant le graphe de Voronoï.

Une attention particulière doit être accordée aux régions non maillées. Comme leur voxels ont été fusionnés, elles ne contiennent ni arêtes, ni faces. Elles formeront à la fin le bord de la 3-carte. Lors du passage au dual, elles seront transformées en des sommets de valences très élevées qui doivent être supprimés. Comme pour les surfaces, les régions obtenues au final ne sont pas toutes des tétraèdres, les sommets cosphériques étant fréquents avec des distances discrètes. Le maillage polyédrique obtenu peut être subdivisé, sans trop difficulté, au besoin.

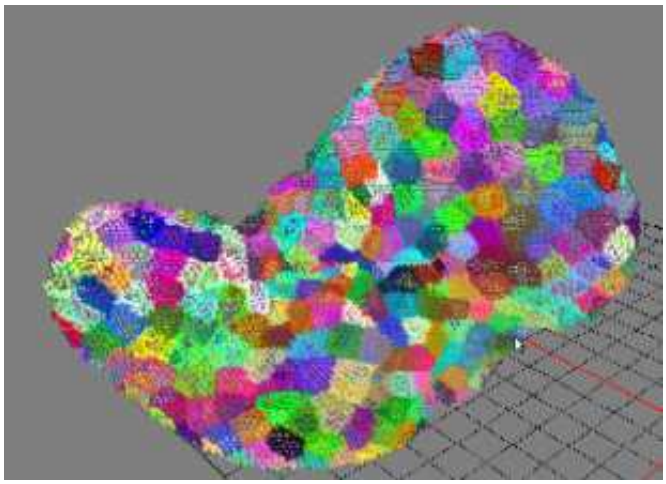


FIGURE 4.15 – Régions du diagramme de Voronoï volumique discret.

4.5 Conclusion

Dans ce chapitre, nous avons présenté la génération simultanée de maillages surfaciques ou volumiques d'objets discrets imbriqués, compatibles entre eux au niveau des zones de contact de ces objets. Basée sur la génération de diagrammes de Voronoï contraints, l'approche discrète présentée est très robuste. Elle s'étend facilement pour obtenir des maillages dont la taille des mailles s'adaptent à la courbure ou à la nature des objets présents.

L'utilisation d'une structure topologique sous-jacente, les micro-3-cartes, permet d'assurer la validité topologique des graphes, et maillages obtenus à toutes les étapes de l'algorithme.

La structure topologique utilisée est définie à différents niveaux hiérarchiques par l'emploi de prédicats encodant de manière implicite une simplification du maillage du niveau inférieur. Cette technique n'est pas liée à la nature discrète des micro-cartes et pourrait s'étendre avec profit, à d'autres algorithmes de simplification de graphes.

Chapitre 5

Projet de recherche

Dans ce mémoire, quatre axes de recherche ont été présentés au coeur desquels les cartes combinatoires jouent un rôle majeur. Le premier axe concernait l'extension du domaine de représentation des cartes pour la modélisation d'objets généraux, dits non variété. Ces derniers sont décomposés en morceaux de variétés, de dimensions éventuellement distinctes, assemblées autour de sommets singuliers. Le second axe de recherche portait sur une extension multirésolution des cartes de dimension 2, sur leur plongement sur des surfaces de subdivision et sur leur utilisation pour l'encodage de maillages progressifs. Pour ces deux extensions, nous avons montré que les modèles obtenus étaient compacts et héritaient naturellement de l'efficacité et de la généricité des cartes.

Les troisième et quatrième axes de recherche concernaient l'utilisation de ces structures combinatoires dans des problèmes d'algorithmique géométrique. Le partitionnement de l'espace en cellules convexes et sa décomposition implicite en tétraèdres orientés nous ont permis de revisiter le problème de la détection de collisions. Nous avons proposé une approche nouvelle à un problème très largement étudié, en obtenant des résultats offrant des perspectives nombreuses. Enfin, les derniers travaux ont montré l'intérêt de structures topologiques fortement contraintes pour la reconstruction ou la simplification de maillages surfaciques et volumiques, en apportant des garanties sur la cohérence des modèles obtenus.

Ces quatre thèmes se retrouvent étroitement liés dans le projet de recherche que je présente ci-dessous, mêlant la spécification, l'étude théorique et l'implantation concrète de structures combinatoires adaptées et spécialisées pour la modélisation, la simulation et le traitement de la géométrie.

De plus en plus d'applications dans ces domaines nécessitent de travailler simultanément avec plusieurs représentations d'un même objet. Celles-ci peuvent correspondre à différentes échelles de visualisation d'une scène ou à différents niveaux de détails d'un même objet pour l'édition multirésolution. Elles peuvent aussi correspondre à des modèles de natures différentes répondant à des besoins applicatifs particuliers. Par exemple, en simulation physique, il n'est pas rare d'utiliser un maillage volumique grossier pour les simulations physiques, auquel un maillage surfacique plus fin est associé pour un rendu réaliste. L'utilisation de représentation multi-échelle est également fréquente dans le domaine de la segmentation d'images, pour la compression, le filtrage ou la simplification de maillages. Enfin de nombreux algorithmes font appel à des structures hiérarchiques pour accélérer les traitements, comme par exemple le lancer de rayon ou la détection de collisions.

Une des difficultés majeures, et à mon sens pas encore suffisamment étudiée, est de comprendre et de traiter correctement les problèmes de communication et de collaboration entre les multiples représentations d'une même réalité. Le coeur de mon projet de recherche est d'aborder ces problèmes en proposant et en étudiant des modèles combinatoires multi-échelles permettant d'englober dans une structure topologique uniforme des maillages de natures, de résolutions ou

de dimensions différentes.

Il s'agira de développer des opérateurs topologiques multi-échelles pour gérer de manière cohérente et contrôlée les interdépendances entre ces représentations. Suivant les applications, ces opérateurs devront permettre de construire de telles hiérarchies, de les manipuler, d'y appliquer des découpes, des simplifications, des raffinements ou d'y faire des requêtes géométriques. Il faudra pouvoir exhiber des propriétés formelles permettant une analyse fine des propriétés de ces modèles et opérateurs, comme c'est historiquement l'usage dans notre équipe.

Avant de pouvoir utiliser ces modèles dans des applications concrètes, il faudra conduire des études précises concernant les coûts en temps et en mémoire des structures et opérateurs proposés. Il faudra aussi s'attacher à développer des librairies logicielles adaptées à différents cadres applicatifs pour assurer la diffusion de nos travaux à l'ensemble de la communauté.

De nombreux projets du LSIIT tournent autour du thème fédérateur de l'imagerie et de la robotique médicale et plus précisément, en ce qui concerne l'équipe IGG et moi-même, autour de la modélisation et de la reconstruction d'objets acquis numériquement et autour de l'utilisation des modèles obtenus pour la visualisation et la simulation d'opérations chirurgicales. C'est donc naturellement que je proposerais des applications et des collaborations dans ce domaine.

Mon programme de recherche, « structures combinatoires pour la modélisation, l'animation et le traitement de la géométrie », s'articule autour de plusieurs axes ayant des perspectives à plus ou moins long termes que je vais détailler ci-dessous.

5.1 Maillages volumiques multirésolutions

L'expérience avec les cartes multirésolutions et les surfaces de subdivisions, nous a montré qu'il était possible de définir des opérations mettant en jeu la géométrie et la topologie à différents niveaux de résolution. A court et moyen terme, je souhaite développer, en suivant ce principe, des modèles volumiques multirésolutions. Ces travaux commencent cette année déjà avec la thèse de Lionel Untereiner que j'encadre avec Dominique Bechmann. Nos voulons proposer des opérateurs topologiques et géométriques permettant de travailler sur des maillages tétraédriques, hexaédriques, puis quelconques avec une approche multi-échelle.

Dans un premier temps il s'agira de proposer et d'étudier un panel complet d'opérateurs de simplification et de raffinement, à la fois topologiques et géométriques, sur de tels maillages. Nous étudierons le coût mémoire de ces modèles combinatoires par rapport aux structures de données utilisées usuellement et testerons les possibilités des opérateurs sur des applications représentatives du domaine.

La première application concerne l'encodage de maillages progressifs volumiques, dans une structure réellement multirésolution. Les opérateurs de contractions sont plus nombreux que dans le cas des surfaces et les contraintes de dégénérescence plus délicates à écrire et gérer. La généricité des cartes devrait nous permettre de proposer des schémas de simplification, plus robustes et d'y mêler des phases d'optimisations permettant de maintenir la qualité géométrique des maillages. Ainsi, des *flips* de tétraèdres modifiant localement la connectivité du maillage sans changer la géométrie des sommets, pourraient être combinés aux fusions d'arêtes, de triangles ou de tétraèdres.

L'approche inverse consiste à générer de nouveaux niveaux de résolution en appliquant des schémas de subdivisions à ces maillages volumiques. Nous chercherons à étendre et généraliser les schémas classiques, comme pour les 2-cartes. Dans ce cadre, deux types d'applications peuvent être visées.

La première concerne le lissage topologique et géométrique du bord de maillages volumiques grossiers. Souvent ce type de maillages est utilisé pour la simulation physique interactive. Pour obtenir un rendu correct, des surfaces leur sont souvent associées, les liens entre les deux maillages étant souvent définis de manière empirique. En subdivisant le bord du maillage volumique, et en le remplissant de volumes de plus en plus fins, sa frontière tendrait vers une surface de subdivision

lisse et naturellement liée au maillage initial. Une approche adaptative, ajustant le niveau de subdivision à la courbure ou au point de vue, permettrais d'obtenir un rendu lisse sans dégrader les performances de la simulation.

La gestion de la connectivité est un point tout aussi important dans ce genre de simulations. Lors de la découpe de l'objet simulé, la surface qui lui est associée doit être découpée également et les différentes surfaces obtenues recollés aux volumes découpés. Ce problème difficile n'est absolument pas résolu dans les systèmes de simulation courants. Les schémas de subdivision proposés le résoudront automatiquement, la connectivité globale des composantes d'une carte étant implicitement définie par les relations topologiques locales.

Les autres applications envisagées pour les maillages volumiques multirésolutions se situent à plus long terme. Elles concernent la subdivision adaptative des volumes pour que les maillages obtenus s'adaptent à un champ de potentiel, une mesure de densité ou à d'autres données volumiques. Ainsi, il serait possible de générer des maillages volumiques adaptés à la discrétisation polyédrique de données volumiques. Des critères de qualité d'une telle construction devront être proposés et comparés avec les travaux existants. De tels modèles pourraient nous permettre d'aborder l'analyse multirésolution de données volumiques, avec des perspectives dans le domaine de la compression de données ou pour l'optimisation de maillages dans le cadre de simulations numériques.

Parallèlement à ces considérations, l'implantation de maillages volumiques progressifs réellement multirésolutions seraient également très utiles pour améliorer ou généraliser différentes méthodes d'affichage ou de rendu volumique, utilisant des niveaux de détails dépendants du point de vue. Ces aspects pourront être approfondis en fonction des résultats obtenus et des projets de l'équipe.

5.2 Partitions de l'espace et hiérarchies volumiques

Des hiérarchies de volumes sont utilisées dans de nombreux travaux en géométrie algorithmique. Elles servent à accélérer certaines requêtes géométriques dans des scènes complexes, comme la localisation de points, l'énumération des éléments proches d'une cible, la recherche d'intersections. C'est le cas, par exemple, dans les algorithmes de rendu par lancers de rayons ou en détection de collisions où il faut savoir si une droite coupe l'une des primitives de la scène ou s'il existe des couples de primitives sécantes.

Dans tous ces cas, la structure hiérarchique est construite, lors d'une phase de prétraitement, puis utilisée telle quelle lors des requêtes géométriques qui suivent. Un des problèmes qui se posent alors est de maintenir à jour cette structure lorsque les éléments de la scène changent. Des méthodes existent qui permettent de gérer correctement le cas de petits déplacements. Par contre, lors de grands déplacements ou lorsque la topologie des objets change, les solutions restent imparfaites et coûteuses en termes de temps de calcul.

Parfois les objets de la scène existent eux-mêmes à différentes résolutions, par exemple dans le cas de simulations temps réel où des modèles physiques multirésolutions sont employées pour réduire les temps de calculs. Dans ce cas, les hiérarchies englobantes sont usuellement construites sur un seul de ces niveaux et souvent le plus détaillé. Il serait souhaitable qu'une hiérarchie de volumes englobant soit compatible et en cohérence avec les niveaux de détails des objets manipulés.

Ces problématiques forment un ensemble de perspectives que je veux poursuivre à plus longs termes. Proposer et construire des hiérarchies volumiques capables de s'adapter aux changements de leurs contenus et à des contenus multi-échelles est un objectif difficile qui pourra être décliné en de nombreux projets.

L'expérience en détection de collision menée durant la thèse de Thomas Jund a démontré que l'on pouvait suivre les déplacements d'un maillage au sein d'une partition volumique de l'espace. Cette partition n'avait qu'un niveau de résolution, mais supportait déjà les déformations et

les changements topologiques. Ces travaux pourraient se poursuivre en utilisant des 3-cartes multirésolutions pour partitionner l'espace.

Lors du suivi des particules, celles-ci peuvent s'enregistrer dans la cellule où elles se trouvent, pour permettre à chaque cellule de maintenir la liste des objets qu'elle contient. En subdivisant les volumes dynamiquement, il serait ainsi possible de limiter le nombre d'objets présents dans chaque cellule. Une telle approche permettrait la gestion des collisions entre objets distincts dans des scènes denses et autoriserait également la détection des auto-collisions, importantes dans le cas d'objets déformables.

En allant plus loin, une partition multirésolution collant naturellement à la géométrie d'une scène permettrait le développement d'algorithmes adaptatifs pour toutes les requêtes géométriques citées précédemment. L'adaptation peut être vu comme un moyen de réduire le nombre de cas à examiner, mais également, pour des applications en temps réel, servir à moduler la précision des réponses (en descendant plus ou moins loin dans la hiérarchie) en fonction du temps imparti au calcul.

Plus généralement, la finalité de cet axe de recherche sera, à long terme, de comprendre comment et à quelles conditions il est possible de maintenir une cohérence topologique entre des éléments de natures différentes, au sein même d'une hiérarchie.

5.3 Opérateurs multi-échelles et cohérence topologique

Ce troisième point concerne des projets à long, voire très long terme, dans le domaine de la modélisation. Les modèles multi-échelles que nous cherchons à construire pourront comporter des données volumiques, surfaciques ou d'autres natures, définies à différents niveaux de détails. Quelle que soit la nature de ces maillages (hiérarchie de volumes englobant, volumes ou surfaces de subdivisions, maillages progressifs, etc.), on doit s'interroger sur les opérateurs topologiques que l'on pourra ou voudra leur appliquer.

Une des questions que nous nous posons est de savoir comment des transformations topologiques effectuées à un niveau de détail donné, peuvent se transmettre à d'autres niveaux, et combien de niveaux de la hiérarchie seront touchés ou autrement dit jusqu'à quelle profondeur la transformation doit être appliquée.

Quelques exemples peuvent illustrer ce propos. Supposons qu'un volume de la hiérarchie doive être subdivisé ainsi que l'une de ses faces. L'opérateur correspondant peut propager cette subdivision aux faces voisines et éventuellement aux volumes adjacents du même niveau de détails. Mais cet opérateur devra également propager cette subdivision aux niveaux de détails plus fins afin qu'ils restent compatibles avec la nouvelle subdivision. Durant cette propagation des changements topologiques peuvent intervenir dans les niveaux plus fins qui devront à leur tour être propagés à leur voisinage dans leur niveau de résolution et aux niveaux inférieurs. A l'opposé de la chaîne, les nouveaux volumes créés par la subdivision initiale doivent être correctement « attaché » aux éléments du niveau supérieur qui les contenaient.

Une autre piste plus ambitieuse concerne la gestion de hiérarchies multiples. Dans la définition actuelle des cartes multirésolutions, les brins sont rangés dans une suite linéaire d'ensembles inclus les uns dans les autres. Il est tout à fait possible de remplacer cette suite, par une arborescence d'ensembles de brins dont les fils contiendraient tous les brins de leurs pères. Un nœud de l'arbre correspondrait à une version du maillage à une résolution donnée, ses fils définissant alors différentes subdivisions possibles. Il serait ainsi possible d'accéder à différentes hiérarchies issues d'un même maillage initial. Elles pourraient être utilisées simultanément, en répondant à des problèmes distincts. Elles fourniraient également la possibilité de comparer ces hiérarchies, en vue par exemple de leur optimisation.

Les contraintes de cohérence topologique et spatiale doivent être très clairement énoncées pour pouvoir être maintenues par ce type d'opérateurs ou de hiérarchies multiples. Il y a là à mon sens des perspectives importantes tant au niveau de la modélisation géométrique que du

génie logiciel. Les structures combinatoires, comme les cartes et les extensions proposées offrent un cadre mathématique et uniforme propice à ce type d'études. D'autre part, il faudra utiliser une méthodologie stricte pour explorer les possibilités et les applications de ces opérateurs. L'expérience en spécifications algébriques que j'ai acquise durant ma thèse, me permet de croire que des résultats intéressants sont atteignables.

5.4 Applications médicales et simulations chirurgicales

Les applications médicales et en particulier la simulation d'opérations chirurgicales sont des applications particulièrement intéressantes pour les projets mentionnés. En effet elles imposent des contraintes fortes sur les modèles géométriques animés : une simulation doit être interactive et donc tous les traitements effectués en temps réel ; elle doit être physiquement réaliste et donc s'appuyer sur des modèles mécaniques précis ; elle doit être visuellement réaliste et donc avoir une géométrie suffisamment fine, mais supportant la détection de collisions et enfin elle doit être spécifique à chaque patient c'est-à-dire ici reconstruite à partir de données réelles fournies par des dispositifs d'imagerie médicale.

Même si dans un premier temps il semble important de travailler d'un point de vue théorique sur les modèles envisagés, les échanges que j'ai eus, avec nos partenaires, lors du projet VORTISS, m'ont permis de proposer à la communauté une implantation de nos modèles topologiques dans le logiciel SOFA (un environnement de développement pour la simulation physique temps réel porté par l'INRIA). Dans la version actuelle de SOFA, les interactions entre représentations sont gérées manuellement par l'utilisateur.

Une des principales difficultés rencontrées est la diversité en terme de structures et de dimension des représentations utilisées : des points pour placer des contraintes géométriques ou des forces ponctuelles (interaction, systèmes de particules pour les modèles *meshless*) ; des éléments linéaires pour la simulation de courbes (fils de suture) ; des éléments surfaciques (masse ressort, différences finies) ou des éléments volumiques (éléments finis, simulation de fluides).

Toutes ces représentations doivent ensuite collaborer entre-elles : la détection d'une collision au niveau grossier, entraîne l'ajout de forces au niveau mécanique, puis des changements au niveau géométrique. Ces collaborations entre différents modèles sont déjà par nature complexes, mais la complexité augmente encore lorsqu'il s'agit de permettre des modifications dans la topologie des maillages. Ceci est bien entendu le cas lorsque l'on souhaite simuler une opération chirurgicale où des tissus peuvent être découpés ou déchirés.

Je souhaite mettre à profit les collaborations déjà existantes pour nourrir ces projets théoriques avec des applications permettant de valider les modèles proposés tant sur le plan des capacités opératoires, que sur celui de l'efficacité.

5.5 Plate-forme CGoGN et diffusion scientifique

Un des rôles des enseignants-chercheurs est de participer à la diffusion des travaux scientifiques qu'ils conduisent. Une part importante de cette diffusion passe bien sur par des communications lors de congrès ou par des publications en revue. Cependant, il me semble important, surtout en informatique, de pouvoir diffuser également nos travaux sous forme de logiciels.

La plate-forme de modélisation CGoGN que je porte depuis quelques années avec Sylvain Thery, ingénieur de recherche dans notre équipe, est un parfait support pour cela. C'est une bibliothèque incluant l'ensemble des modèles, opérateurs et algorithmes développés dans l'équipe. C'est un outil important pour diffuser nos travaux après de partenaires non spécialistes en modélisation géométrique, ou auprès d'industriels.

La plate-forme CGoGN est en cours d'intégration avec la plate-forme SOFA ce qui assurera la diffusion de nos modèles et approches topologiques à l'ensemble de la communauté de simulation

ou de mécaniques.

Les travaux proposés ici seront implantés dans cette plate-forme, ainsi que les autres travaux à venir de l'équipe. Je m'attacherai à proposer cette plate-forme à des équipes de chercheurs ou à des entreprises partenaires afin de diffuser les connaisances qui y seront accumulées et de valider sur des problématiques différentes les modèles et algorithmes développés.

5.6 Conclusion

Le présent programme de recherche nécessite des compétences en modélisation géométrique, en topologie combinatoire, et plus largement en informatique graphique (animation, simulation, acquisition et visualisation) et en mathématiques appliquées. Ces compétences existent en partie au LSIIT à Strasbourg et dans l'équipe IGG, mais devront également être trouvées chez des partenaires avec lesquels des collaborations sont en cours, ou auprès de nouveaux partenariats.

Bibliographie

[1] P. Alliez and C. Gotsman. *Recent advances in compression of 3d meshes*, pages 3–26. Springer-Verlag, 2005.

[2] S. Ansaldi, L. De Floriani, and B. Falcidieno. Geometric modeling of solid objects by using a face adjacency graph representation. *ACM SIGGRAPH Computer Graphics*, 19(3) :131–139, 1985.

[3] D. Arquès and P. Koch. Modélisation de solides par les pavages. In *Proceedings of Pixim*, pages 47–61, Paris, 1989.

[4] D. Badouel and G. Hégron. An evaluation of CSG trees based on polyhedral solids. In *Proceedings of Eurographics*, Nice, 1988.

[5] F. Baldacci, A. Braquelaire, P. Desbarats, and J.-P. Domenger. 3d image topological structuring with an oriented boundary graph for split and merge segmentation. In *DGCI'08 : proceedings of the international Conference on Discrete Geometry for Computer Imagery*, pages 541–552, Berlin, Heidelberg, 2008. Springer-Verlag.

[6] B.G. Baumgart. A polyhedron representation for computer vision. In *Proceedings of AFIPS*, volume 44 of *National Computer Conference*, pages 589–596, 1975.

[7] E. Bertin. *Diagrammes de Voronoï 2D et 3D : Applications en analyse d'images.* PhD thesis, Univeristé Joseph Fourier, Grenoble, 1994.

[8] M. Bertram, G. Reis, R.H. Van Lengen, S. Köhn, and H. Hagen. Non-manifold mesh extraction from time-varying segmented volumes used for modeling a human heart. In *Eurovis 05*, pages 1–10, 2005.

[9] Y. Bertrand, G. Damiand, and C. Fiorio. Topological encoding of 3d segmented images. In *DGCI'00 : proceedings of the international Conference on Discrete Geometry for Computer Imagery*, pages 311–324, 2000.

[10] Daniel Bielser, Pascal Glardon, Matthias Teschner, and Markus Gross. A state machine for real-time cutting of tetrahedral meshes. *11th Pacific Conference on Computer Graphics and Applications*, page 377, 2003.

[11] H. Biermann, D. Kristjansson, and D. Zorin. Approximate boolean operations on free-form solids. In *Proceedings of SIGGRAPH'01*, pages 185–194, 2001.

[12] H. Biermann, A. Levin, and D. Zorin. Piecewise smooth subdivision surfaces with normal control. In *Proceedings of SIGGRAPH'00*, pages 185–194, 2000.

[13] H. Biermann, I. Martin, F. Bernardini, and D. Zorin. Cut-and-paste editing of multiresolution surfaces. *ACM Transactions on Graphics*, Special issue for SIGGRAPH conference 21(3) :312–321, 2002.

[14] S. Bischoff and L. Kobbelt. Extracting consistent and manifold interfaces from multi-valued volume data sets. In *Bildverarbeitung für die Medizin*, 2006.

[15] D. Boltcheva. *Modélisation géométrique et topologique des images discrètes.* PhD thesis, LSIIT-ULP, october 2007.

[16] D. Boltcheva and D. Bechmann. Discrete delaunay : Boundary extraction from voxel objects. In *6th International Conference on 3-D Digital Imaging and Modeling*. IEEE Computer Society Press, august 2007.

[17] D. Boltcheva, D. Bechmann, D. Cazier, C. Kern, S. Thery, and P. Schreck. Reconstruction multi-objets d'images 3d multi-labels à partir d'un algorithme de delaunay discret. *Journal REFIG (Revue Electronique Francophone d'Informatique Graphique)*, 3(1) :53–65, 2009.

[18] G. Borgefors. Distance transformations in digital images. *Computer Vision Graphics and Image Processing*, 34 :344–371, 1986.

[19] R.P. Brian and D. Singerman. Foundation of the theory of maps on surfaces with boundary. *Quart. journal of Math. Oxford*, pages 17–41, 1985.

[20] E. Brisson. Representing geometry structures in d dimensions : Topology and order. *Discrete & Computational Geometry*, pages 387–426, 1993.

[21] Swen Campagna, Leif Kobbelt, and Hans-Peter Seidel. Directed edges - a scalable representation for triangle meshes. *journal of Graphics Tools*, 3(4) :1–11, 1998.

[22] E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6) :350–355, 1978.

[23] D. Cazier and J.-F. Dufourd. A rewrite system to build planar subdivisions. In *Proceedings of the seventh Canadian Conference on Computational Geometry*, pages 235–240, august 1995.

[24] D. Cazier and J.-F. Dufourd. Rewriting-based derivation of efficient algorithms to build planar subdivisions. In *Spring Conference on Computer Graphics*, pages 45–54, june 1996.

[25] D. Cazier and J.-F. Dufourd. Reliable boolean operations on polyhedral solids thanks to a 3d refinement. In *Winter School on Computer Graphics*, pages 40–49, Plzen, Czech Republic, february 1997.

[26] D. Cazier and J.-F. Dufourd. Term rewrite systems to derive set boolean operations on 2d-objects. In *Formal Methods Europe*, volume 1313 of *Lecture Notes in Computer Science*, pages 605–623. Springer Verlag, 1997.

[27] D. Cazier and J.-F. Dufourd. A formal specification of geometric refinements. *The Visual Computer Journal*, 15 :279–301, 1999.

[28] D. Cazier and P. Kraemer. X-maps : an efficient model for non-manifold modeling. In *Shape Modeling International (SMI'10)*. IEEE Conference Publishing Services, june 2010.

[29] David Cazier. *Construction de Système de réécriture pour les opérations booléennes en modélisation géométrique*. PhD thesis, Université Louis Pasteur, Strasbourg, 1997.

[30] David Cazier and Sylvain Thery. CGoGN : Combinatorial and geometric modeling with generic n-dimensional maps. Technical report, LSIIT, Université de Strabourg, CNRS, https ://iggservis.u-strasbg.fr/CGoGN, 2010.

[31] W.W. Charlesworth and D.C. Anderson. Applications of non-manifold topology. In *Proceedings of Computers in Engineering Conference and Engineering Database Symposium*, pages 103–112, 1995.

[32] S.W. Choi, T.K. Dey, I.H. Son, and Y.T. Im. Tetrahedral mesh generation based on advancing front technique and optimization scheme. *International journal for Nimerical Methods in Engineering*, 58 :1857–1872, 2003.

[33] Jonathan D. Cohen, Ming C. Lin, Dinesh Manocha, and Madhav Ponamgi. I-collide : an interactive and exact collision detection system for large-scale environments. In *SI3D '95 : Proceedings of the 1995 symposium on Interactive 3D graphics*, page 189. ACM Press, 1995.

[34] R. Cori. Un code pour les graphes planaires ses applications. *Astérisque*, 27 :164–168, 1975.

[35] R. Cori and A. Machí. Maps, hypermaps and their automorphisms, a survey. *Expositiones Mathematicae*, 10 :403–467, 1992.

[36] Robert Cori. Hypermaps and indecomposable permutations. *European Journal of Combinatorics*, 30(2) :540–541, 2009.

[37] G.A. Croker and W.F. Reinke. An editable nonmanifold boundary representation. *IEEE Computer Graphics & Applications*, pages 39–51, 1991.

[38] G. Damiand. Topological model for 3d image representation : Definition and incremental extraction algorithm. *Computer Vision and Image Understanding*, 109(3) :260–289, 2008.

[39] G. Damiand, M. Dexet, P. Lienhardt, and E. Andres. Removal and contraction operations to define combinatorial pyramids : application to the design of a spatial modeler. *Image and Vision Computing*, 23(2) :259–269, 2005.

[40] Leila De Floriani, David Greenfieldboyce, and Annie Hui. A data structure for non-manifold simplicial d-complexes. In *SGP'04 : Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 83–92, 2004.

[41] Leila De Floriani and Annie Hui. A scalable data structure for three-dimentional non-manifold objects. In *SGP '03 : Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 72–82, 2003.

[42] Leila De Floriani and Annie Hui. Data structures for simplicial complexes : an analysis and a comparison. In *SGP '05 : Proceedings of the third Eurographics symposium on Geometry processing*, page 119, Vienna, Austria, 2005.

[43] L. DeFloriani, L. Kobbelt, and E. Puppo. A survey on data structures for level-of-detail models. *Advances in Multiresolution for Geometric Modelling, series in Mathematics and Visualization*, pages 49–74, 2004.

[44] L. DeFloriani, P. Magillo, and E. Puppo. Efficient implementation of multi-triangulations. In *IEEE Visualization*, pages 43–50, 1998.

[45] L. DeFloriani and E. Puppo. Hierarchical triangulation for multiresolution surface description. *ACM Transactions on Graphics*, 14(4) :363–411, 1995.

[46] D.P. Dobkin and M.J. Laszlo. Primitives for the manipulation of three-dimensional subdivisions. In *Proceedings of 3rd ACM Symposium on Computional Geometry*, pages 86–99, 1987.

[47] D. Doo. A subdivision algorithm for smoothing down irregularly shaped polygons. In *Proceedings on Interactive Techniques in Computer Aided Design*, pages 157–165, 1978.

[48] D. Doo and M. Sabin. Analysis of the behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10(6) :356–360, 1978.

[49] J.F. Dufourd. Algebraic map-based topological kernel for polyhedron modellers : algebraic specification and logic prototyping. In *Proceedings of Eurographics*, pages 649–662, 1989.

[50] J.F. Dufourd. Formal specification of topological subdivisions using hypermaps. *Computer-Aided Design*, 23(2) :99–116, 1991.

[51] N. Dyn, D. Levin, and J. A. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Trans. Graph.*, 9(2) :160–169, 1990.

[52] J. Edmonds. A combinatorial representation for polyhedral surfaces. In *Notices American Mathematical Society*, volume 7, 1960.

[53] H. Elter and P. Lienhardt. Cellular complexes as structured semi-simplicial sets. *International journal of Shape Modeling*, 1(2) :191–217, 1995.

[54] François Faure, Sébastien Barbier, Jérémie Allard, and Florent Falipou. Image-based collision detection and response between arbitrary volumetric objects. In *ACM Siggraph/Eurographics Symposium on Computer Animation, SCA 2008, July, 2008*, pages 155–162, 2008.

[55] J. Françon. Géométrie discrète et algorithmique graphique, problèmes et solutions. In *Actes des Journées AFCET*, Nice, 1988. Groplan.

[56] Bernhard Geiger. Real-time collision detection and response for complex environments. In *CGI'00 : Proceedings of the International Conference on Computer Graphics*, page 105. IEEE Computer Society, 2000.

[57] S.F.F Gibson. Constrained elastic surface nets : Generating smooth surfaces fro mbinary segmented data. In *MICCAI*, pages 888–898, 1998.

[58] E.G. Gilbert, D.W. Johnson, and S.S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *Robotics and Automation, IEEE Journal of [see also IEEE Transactions on Robotics and Automation]*, 4(2) :193–203, 1988.

[59] S. Gottschalk, M. C. Lin, and D. Manocha. Obbtree : a hierarchical structure for rapid interference detection. In *SIGGRAPH'96 : Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 171–180. ACM Press, 1996.

[60] Naga K. Govindaraju, David Knott, Nitin Jain, Ilknur Kabul, Rasmus Tamstorf, Russell Gayle, Ming C. Lin, and Dinesh Manocha. Interactive collision detection between deformable models using chromatic decomposition. *ACM Transaction on Graphics*, 24(3) :991–999, 2005.

[61] C. Grasset-Simon, G. Damiand, and P. Lienhardt. nD generalized map pyramids : definition, representations and basic operations. *Pattern Recognition*, 39(4) :527–538, 2006.

[62] Alexander Greß, Michael Guthe, and Reinhard Klein. Gpu-based collision detection for deformable parameterized surfaces. *Computer Graphics Forum*, 25(3) :497–506, 2006.

[63] S. Gueorguieva and D. Marcheix. Non-manifold boundary representation for solid modeling. In *Proceedings International Computer Symposium*, 1994.

[64] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoï diagrams. *ACM Transactions on Graphics*, 4(2) :74–123, April 1985.

[65] E.L. Gursoz, Y. Choi, and F.B. Printz. Vertex-based representation of non-manifold boundaries. *Geometric Modeling for Product Engineering*, pages 107–130, 1990.

[66] E.L. Gursoz, Y. Choi, and F.B. Printz. Boolean set operations on non-manifold boundary representation objects. *Computer-Aided Design*, 23(1) :33–39, 1991.

[67] Stéphane Guy and Gilles Debunne. Monte-carlo collision detection. Technical Report RR-5136, INRIA, 2004.

[68] H.-C. Hege, M. Seebass, D. Stalling, and M. Zökler. A generalized marching cubes algorithm based on non-binary classification. Technical report, Konrad-Zuse-Zentrum, 1997.

[69] Y. Hijazi, D. Bechmann, D. Cazier, C. Kern, and S. Thery. Fully-automatic branching reconstruction algorithm : application to vascular trees. In *Shape Modeling International (SMI'10)*. IEEE Conference Publishing Services, june 2010.

[70] C.M. Hoffmann, J.E Hopcroft, and M.S. Karasick. Robust set operations on polyhedral solids. *IEEE Computer Graphics & Applications*, 9(6) :50–59, 1989.

[71] H. Hoppe. Progressive meshes. In *Proceedings of SIGGRAPH'96*, pages 99–108, 1996.

[72] H. Hoppe. Efficient implementation of progressive meshes. *Computer & Graphics*, 22(1) :27–36, 1998.

[73] Philip M. Hubbard. Collision detection for interactive graphics applications. *IEEE Transactions on Visualization and Computer Graphics*, 1(3) :218–230, 1995.

[74] A. Jacques. Constellations et graphes topologiques. In *Colloque Math. Soc.*, pages 657–672, 1970.

[75] P. Jiménez, F. Thomas, and C. Torras. 3D Collision Detection : A Survey. *Computers and Graphics*, 25(2) :269–285, 2001.

[76] Lionel Joussemet. *Approche évolutionniste pour la détection des collisions au sein d'environnements virtuels denses*. PhD thesis, Université Montpellier II, 2006.

[77] T. Jund, D. Cazier, and J.-F. Dufourd. Particle-based forecast mechanism for continuous collision detection in deformable environments. In *SIAM/ACM Joint Conference on Geometric and Physical Modeling*, pages 147–158. ACM, october 2009.

[78] T. Jund, D. Cazier, and J.-F. Dufourd. Système de prédiction pour la détection de collisions dans un environnement déformable. *Journal REFIG (Revue Electronique Francophone d'Informatique Graphique)*, 3(2) :47–58, 2009.

[79] T. Jund, D. Cazier, and J.-F. Dufourd. Edge collision detection in complex deformable environments. In *Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS)*. Eurographics Association, november 2010.

[80] T. Jund, D. Cazier, and J.-F. Dufourd. Edge collision detection in complex environment. In *Computer Graphics International*, june 2010.

[81] Thomas Jund. *Détection de collisions dans des environements complexes et déformables*. PhD thesis, Université de Strasbourg, septembre 2010.

[82] James T. Klosowski, Martin Held, Joseph S.B. Mitchell, Henry Sowizral, and Karel Zikan. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics*, 4(1) :21–36, 1998.

[83] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of SIGGRAPH'98*, pages 105–114, 1998.

[84] P. Kraemer, D. Cazier, and D. Bechmann. Extension multirésolution des cartes combinatoires : application à la représentation de surfaces de subdivision multirésolution. *Journal REFIG (Revue Electronique Francophone d'Informatique Graphique)*, 1(1) :21–32, march 2007. taux d'acception 25

[85] P. Kraemer, D. Cazier, and D. Bechmann. A general and efficient representation for multiresolution meshes : application to quad/triangle subdivision. In *Proceedings of CCCG'07 (Canadian Conference on Computational Geometry)*, pages 257–260, august 2007.

[86] P. Kraemer, D. Cazier, and D. Bechmann. Multiresolution half-edges. In *Proceedings of SCCG'07 (Spring Conference on Computer Graphics)*, pages 242–249. ACM Press, april 2007.

[87] P. Kraemer, D. Cazier, and D. Bechmann. Extension of half-edges for the representation of multiresolution subdivision surfaces. *The Visual Computer*, 25(2) :149–163, 2009.

[88] Pierre Kraemer. *Modèles topologiques pour la multirésolution*. PhD thesis, Université Louis Pasteur, Strasbourg, november 2008.

[89] Walter G. Kropatsch. Abstraction pyramids on discrete representations. In *DGCI'02 : proceedings of the international Conference on Discrete Geometry for Computer Imagery*, pages 1–21, 2002.

[90] François Labelle and Jonathan Richard Shewchuk. Isosurface stuffing : fast tetrahedral meshes with good dihedral angles. *ACM Transactions on Graphics*, 26(3) :57, 2007.

[91] V. Lang and P. Lienhardt. Geometric modeling with simplicial sets. In *IEEE Computer Graphics & Applications*, Proceedings of Pacific Graphics'95, pages 475–494, 1995.

[92] M. Lee and H. Samet. Navigating through triangle meshes implemented as linear quadtrees. *ACM Transactions on Graphics*, 19(2) :79–121, 2000.

[93] S. H. Lee. Offstetting operations in non-manifold geometric modeling. In *Proceedings of the 5th ACM Symposium on Solid Modeling and Applications*, pages 42–53, Ann Arbor, Michigan, 1999.

[94] S. H. Lee and K. Lee. Partial entity structure : A compact non-manifold boundary representation based on partial topological entities. In *Proceedings Symposium on Solid Modeling and Applications*, pages 159–170, 2001.

[95] Julien Lenoir, Stephane Cotin, Christian Duriez, and Paul F. Neumann. Interactive physically-based simulation of catheter and guidewire. *Computers & Graphics*, 30(3) :416–422, 2006.

[96] P. Lienhardt. Subdivision of n-dimensional spaces and n-dimensional generalized maps. In *5th ACM Conf. on Computational Geometry*, pages 228–236, Saarbrücken, Germany, 1989.

[97] P. Lienhardt. Topological models for boundary representation : a comparison with *n*-dimensional generalized maps. *Computer-Aided Design*, 23(1) :59–82, 1991.

[98] M.C. Lin and J.F. Canny. A fast algorithm for incremental distance calculation. *Proceedings of the IEEE International Conference on Robotics and Automation.*, 2 :1008–1014, 1991.

[99] Ming C. Lin and Dinesh Manocha. *Handbook of Discrete and Computational Geometry*, chapter 35 - Collision and proximity queries. CRC Press, 2004.

[100] Alex Lindblad and George Turkiyyah. A physically-based framework for real-time haptic cutting and interaction with 3d continuum models. In *SPM'07 : Proceedings of the 2007 ACM symposium on Solid and physical modeling*, pages 421–429, 2007.

[101] C. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, 1987.

[102] M. Lounsberry, T. DeRose, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics*, 16(1) :34–73, 1997.

[103] Y. Luo. Generalized euler operators for non-manifold boundary solid modeling. In *Geometric Modelling Studies 1990/3*, pages 19–34. MTA SZTAKI, Hungary, 1993.

[104] D. Marcheix and S. Gueorguieva. Topological operators for non-manifold modeling. In *Proceedings of the 30th International Symposium on Automotive Technology and Automation, Mechatronics/Automotive Electronics*, pages 173–186, Florence, Italy, 1997.

[105] H. Masuda. Topological operators and boolean operations for complex-based nonmanifold geometric models. *Computer-Aided Design*, 25(2) :119–129, 1992.

[106] M. Mäntylä. Boolean set operations on 2-manifolds through vertex neihborhood. *Transaction on Graphics*, 5(1) :1–29, 1986.

[107] M. Mäntylä and R. Sulonen. GWB : a solid modeler with Euler operators. *IEEE Computer Graphics & Applications*, 2(7) :17–31, 1982.

[108] Démian Nave, Nikos Chrisochoides, and L. Paul Chew. Guaranteed-quality parallel delaunay refinement for restricted polyhedral domains. *Computational Geometry : Theory and Applications*, 28(2-3) :191–215, 2004.

[109] G.M. Nielson. Dual marching cubes. In *VIS'04 : Proceedings of the conference on Visualization*, pages 489–496, 2004.

[110] Miguel A. Otaduy and Ming C. Lin. Clods : dual hierarchies for multiresolution collision detection. In *SGP'03 : Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 94–101, 2003.

[111] A. Paoluzzi, F. Bernardini, C. Cattani, and V. Ferrucci. Dimension-independent modeling with simplicial complexes. *ACM Transactions on Graphics*, 12(1) :56–102, 1993.

[112] S. Pesco, G. Tavares, and H. Lopes. A stratification approach for modeling two-dimensional cell complexes. *Computers & Graphics*, 28(2) :235–247, 2004.

[113] J. Peters and U. Reif. The simplest subdivision scheme for smoothing polyhedra. *ACM Trans. Graph.*, 16(4), 1997.

[114] J. Popovic and H. Hoppe. Progressive simplicial complexes. In *Proceedings of SIGGRAPH'97*, pages 99–108, 1997.

[115] L.K. Putnam and P.A. Subrahmanyam. Boolean operations on n-dimensional objects. *IEEE Computer Graphics & Applications*, 6(6) :43–51, 1986.

[116] B. Reitinger, A. Bornik, and R. Beichel. Constructing smooth non-manifold meshes of multi-labeled volumetric datasets. In *Winter School on Computer Graphics*, pages 227–234, 2005.

[117] A.A.G. Requicha. Representation for rigid solids : theory, methods and systems. *Computing Survey*, 12(4) :437–463, 1980.

[118] A.A.G. Requicha and H.B. Voelcker. Boolean operations in solid modeling : boundary evaluation and merging algorithms. In *Proceedings of IEEE*, volume 73, 1985.

[119] J. Rossignac, A. Safonova, and A. Szymczak. 3d compression made simple : Edgebreaker on a corner table. In *Proceedings of Shape Modeling International*, pages 278–283, Genova, Italy, 2001.

[120] J.R. Rossignac and M.A O'Connor. SGC : A dimension-independent model for pointsets with internal structures and incomplete boundary. *Computer-Aided Design*, 1991.

[121] G. Schrack. Finding neighbors of equal size in linear quadtrees and octrees in constant time. *CVGIP : Image Understanding*, 55(3) :221–230, 1992.

[122] V. Shapiro. Solid modeling. In G. Farin, J. Hoschek, and M.S. Kim, editors, *Handbook of computer aided geometric design*, chapter 20, pages 473–518. Elsevier Science Publishers, Amsterdam, 2002.

[123] J.R. Shewchuk. Tetrahedral mesh generation by delaunay refinement. In *SCG'8 : Proceedings on Computational Geometry*, pages 86–95, 1998.

[124] L. Soler, N. Ayache, S. Nicolau, X. Pennec, C. Forest, H. Delingette, D. Mutter, and J.Marescaux. Traitements d'images medicales pour la planification, la simulation et l'aide intra-opératoire des actes chirurgicaux. In M. Faupel, P. Smigielski, and R.Grzymala, editors, *Imagerie et Photonique pour les sciences du vivant et la medecine*, pages 19–31. edition Fontis Media, 2004.

[125] J.C. Spehner. Merging in maps and in pavings. *Theoretical Computer Science*, pages 205–232, 1991.

[126] G. Taubin. A signal processing approach to fair surface design. In *Proceedings of SIGGRAPH'95*, pages 351–358, 1995.

[127] M. Teschner, S. Kimmerle, G. Zachmann, B. Heidelberger, Laks Raghupathi, A. Fuhrmann, Marie-Paule Cani, François Faure, N. Magnetat-Thalmann, and W. Strasser. Collision detection for deformable objects. In *Eurographics State-of-the-Art Report (EG-STAR)*, pages 119–139. Eurographics Association, 2004.

[128] R.B. Tilove. Set membership classification : A unified approach to geometric intersection problems. *IEEE Trans. Computer*, 29(10) :874–883, 1980.

[129] Jane Tournois, Camille Wormser, Pierre Alliez, and Mathieu Desbrun. Interleaving delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Transactions on Graphics*, 28(3) :1–9, 2009.

[130] W. Tutte. Graph theory. In *Encyclopedia of Mathematics and its Applications*, chapter 21. Cambridge University Press, 1984.

[131] Andrew Vince. Regular combinatorial maps. *journal of Combinatorial Theory*, 35(3) :256–277, 1983.

[132] K. Weiler. Edge-based data structures for modeling in curved-surface environments. *IEEE Computer Graphics & Applications*, 5(1) :21–40, 1985.

[133] Kevin Weiler. Polygon comparison using a graph representation. In *SIGGRAPH'80 : Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, pages 10–18, 1980.

[134] K.J. Weiler. Boundary graphs operators for non-manifold geometric modeling topology representations. *Geometric Modeling for CAD Applications*, pages 37–66, 1988.

[135] K.J. Weiler. The radial edge structure : a topological representation for non-manifold geometric modeling. *Geometric Modeling for CAD Applications*, pages 3–36, 1988.

[136] K.J. Weiler. Generalized sweep operations in the non-manifold environment. In *Geometric modeling for product engineering*. Elsevier, 1990.

[137] Z. Wu and J. M. Sullivan. Multiple material marching cubes algorithm. *International journal for numerical methods in engineering*, pages 189–207, 2003.

[138] Y. Yamaguchi and F. Kimura. Nonmanifold topology based on coupling entities. *IEEE IEEE Computer Graphics & Applications*, 15(1) :42–50, 1995.

[139] D. Zorin. Modeling with multiresolution subdivision surfaces. *Tutorial Eurographics*, 2005.

[140] D. Zorin, P. Schröder, and W. Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *Proceddings of SIGGRAPH'96*, pages 189–192, 1996.

[141] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. In *Proceedings of SIGGRAPH'97*, pages 259–268, 1997.

Annexe A

Publications

Ouvrages

D. Cazier and C. Minich. *Informatique Graphique, modélisation géométrique et animation*, chapter Modélisation géométrique : les modèles classiques, pages 23–47. Traité IC2. Hermès, march 2007.

Communications dans des revues internationales et nationales avec comités de lecture

P. Kraemer, D. Cazier, and D. Bechmann. Extension of half-edges for the representation of multiresolution subdivision surfaces. *The Visual Computer*, 25(2) :149–163, 2009.

D. Cazier and J.-F. Dufourd. A formal specification of geometric refinements. *The Visual Computer Journal*, 15 :279–301, 1999.

D. Boltcheva, D. Bechmann, D. Cazier, C. Kern, S. Thery, and P. Schreck. Reconstruction multi-objets d'images 3d multi-labels à partir d'un algorithme de delaunay discret. *Journal REFIG (Revue Electronique Francophone d'Informatique Graphique)*, 3(1) :53–65, 2009.

T. Jund, D. Cazier, and J.-F. Dufourd. Système de prédiction pour la détection de collisions dans un environnement déformable. *Journal REFIG (Revue Electronique Francophone d'Informatique Graphique)*, 3(2) :47–58, 2009.

P. Kraemer, D. Cazier, and D. Bechmann. Extension multirésolution des cartes combinatoires : application à la représentation de surfaces de subdivision multirésolution. *Journal REFIG (Revue Electronique Francophone d'Informatique Graphique)*, 1(1) :21–32, march 2007. taux d'acception 25%.

Communications dans des conférences internationales avec comités de lecture

T. Jund, D. Cazier, and J.-F. Dufourd. Edge collision detection in complex environment. In *Computer Graphics International*, june 2010.

D. Cazier and P. Kraemer. X-maps : an efficient model for non-manifold modeling. In *Shape Modeling International (SMI'10)*. IEEE Conference Publishing Services, june 2010.

Y. Hijazi, D. Bechmann, D. Cazier, C. Kern, and S. Thery. Fully-automatic branching reconstruction algorithm : application to vascular trees. In *Shape Modeling International (SMI'10)*. IEEE Conference Publishing Services, june 2010.

T. Jund, D. Cazier, and J.-F. Dufourd. Particle-based forecast mechanism for continuous collision detection in deformable environments. In *SIAM/ACM Joint Conference on Geometric and Physical Modeling*, pages 147–158. ACM, october 2009.

P. Kraemer, D. Cazier, and D. Bechmann. A general and efficient representation for multiresolution meshes : application to quad/triangle subdivision. In *Proceedings of CCCG'07 (Canadian Conference on Computational Geometry)*, pages 257–260, august 2007.

P. Kraemer, D. Cazier, and D. Bechmann. Multiresolution half-edges. In *Proceedings of SCCG'07 (Spring Conference on Computer Graphics)*, pages 242–249. ACM Press, april 2007.

D. Cazier and J.-F. Dufourd. Term rewrite systems to derive set boolean operations on 2d-objects. In *Formal Methods Europe*, volume 1313 of *Lecture Notes in Computer Science*, pages 605–623, Gratz (Austria), 1997. Springer Verlag.

D. Cazier and J.-F. Dufourd. Reliable boolean operations on polyhedral solids thanks to a 3d refinement. In *Winter School on Computer Graphics*, pages 40–49, Plzen (Czech Republic), february 1997.

D. Cazier and J.-F. Dufourd. Rewriting-based derivation of efficient algorithms to build planar subdivisions. In *Spring Conference on Computer Graphics*, pages 45–54, Bratislava (Slovakia), june 1996.

D. Cazier and J.-F. Dufourd. A rewrite system to build planar subdivisions. In *Proceedings of the seventh Canadian Conference on Computational Geometry*, pages 235–240, Université Laval (Québec), august 1995.

Communications dans des manifestations d'audiences nationales

T. Jund, D. Cazier, and J.-F. Dufourd. Système de prédiction pour la détection de collisions dans un environnement déformable. In *Journées AFIG 2008*, pages 13–22, Toulouse, november 2008.

P. Kraemer, D. Cazier, and D. Bechmann. Un modèle générique pour la manipulation de maillages multirésolution. In *Journées AFIG 2007*, pages 41–48, Marne-la-Vallée, november 2007.

T. Jund, D. Cazier, and D. Bechmann. Navigation 3d virtuelle : positionnement et technique de sélection dans un affichage volumétrique complexe. In *Journées AFIG 2007*, pages 91–98, Marne-La-Vallée, november 2007.

P. Kraemer, D. Cazier, and D. Bechmann. Extension multirésolution des cartes combinatoires : application à la représentation de surfaces de subdivision multirésolution. In *Journées AFIG 2006*, pages 19–26, Bordeaux, november 2006.

Thèse

David Cazier. *Construction de Système de réécriture pour les opérations booléennes en modélisation géométrique.* Thèse de doctorat, Université L. Pasteur, Strasbourg, 1997.

Thèses co-encadrées

Thomas Jund. *Détection de collisions dans des environements complexes et déformables.* Thèse de doctorat, Université de Strasbourg, septembre 2010.

Pierre Kraemer. *Modèles topologiques pour la multirésolution.* Thèse de doctorat, Université Louis Pasteur, Strasbourg, november 2008.

Annexe B

Reconstruction de Vaisseaux

Fully-automatic branching reconstruction algorithm: application to vascular trees

Younis Hijazi*†, Dominique Bechmann*, David Cazier*, Cyril Kern* and Sylvain Thery*
*IGG-LSIIT, UMR 7005 CNRS - UdS, University of Strasbourg, France
Email: {hijazi, bechmann, david.cazier, kern, thery}@unistra.fr
†Fraunhofer-ITWM and University of Kaiserslautern, Kaiserslautern, Germany
Email: hijazi@itwm.fhg.de

***Abstract*—Reconstructing tubular structures with high-order branching is a difficult task to perform automatically. Medical applications in particular demand accurate models of such objects that fulfill specific topological and geometric criteria. Indeed, the reconstructed object should be a 2-manifold surface with compact, adaptive geometry. We present a generic algorithm for automatically reconstructing n-furcated tubular surfaces. Our approach relies on a strong underlying topological structure and a novel n-furcation reconstruction algorithm using convex entities.**

***Keywords*-reconstruction; branching; furcation; automatic; topology; half-edge; modeling; vascular.**

I. Introduction

Tree structures are everywhere in nature. These structures can be very complex and have a high degree of branching. For example a vascular tree can have up to seven simultaneous branches. Reconstructing the tubular structure itself isn't the challenge; modeling the intersection regions is the difficult task. The input of our algorithm is a tree where each point is attached to a radius. Our test data, vascular trees from the liver, were provided by the medical research institute IRCAD in Strasbourg. Our goal is to fully-automatically reconstruct, from these data, a topologically and geometrically consistent 2-manifold mesh of a tubular structure. This mesh may then be used in simulators for surgery planning, e.g. in a virtual/augmented reality system.

When concerned with the reconstruction of medical tubular structures such as blood vessels there are two main approaches: the model-based reconstruction and the model-free reconstruction. The latter aims at reconstructing the surface directly from volume data e.g. CT-scans; the main algorithm for this purpose is the marching cubes [1] but there are also more sophisticated variants such as the Multi-level Partition of Unity Implicits [2]. In our paper we will only consider model-based reconstruction, i.e. reconstruction based on a tree.

Among recent surveys on vasculature reconstruction Kirbas et al. [3] review vessel extraction techniques and algorithms whereas Oeltze [4] reviews the most common approaches for the visualization of vasculature.

II. Related Work

One of the first methods for reconstructing tubular structures were based on cylinder fitting [5] and freeform surfaces [6]. Then came truncated cone fitting [7] in order to better represent the data. Implicit surfaces were successfully applied to the modeling of tree surfaces, e.g. by Bloomenthal to model the Mighty Maple [8], but these methods require the tuning of a blending function, which is a difficult task [9]. More recently, convolution surfaces (generalization of implicit surfaces) were applied to the reconstruction of vascular structures [4], [10], [11]. They provide high quality surfaces but are complex and computationally expensive. Other methods based on Delaunay triangulation provide good results for mesh reconstruction [12], [13] but have not been specifically demonstrated for complex vascular structures. Felkel et al. [14], [15] and Ou et al. [16] applied subdivision surfaces. Felkel's algorithm is fast and simple but not artifact-free and Ou's algorithm is general but not robust for all scenarios. Cai et al. [17] used a constructive approach which can only handle bifurcations. Lluch et al. [18] applied L-systems to plant and tree modeling and Tian et al. [19]

applied generalized cylinders for plant roots. Both methods have only been demonstrated on low-degree branching. Bornik et al. [20] developed a two-steps algorithm based on simplex meshes and an iterative mesh deformation but the paper doesn't provide enough details for comparison. Zhan et al. [21] applied isogeometric analysis and NURBS and obtained excellent results but the algorithm lacks of genericity as it relies on specific templates. Another family of mesh reconstruction algorithms are manifold meshes [22]–[26]. The algorithms presented by George Hart [22], [26] and by Mandal et al. [24] (column modeling) are quite similar to the algorithm presented in this paper but differ in their applications. George Hart uses a set of lines as a starting point. In column modeling, the edges of a manifold mesh are converted to a column, which can be a tree structure. George Hart's method and column modeling do not allow changing the cross-section whereas, in our paper, the cross-section can change by giving a nice structure. Our work is motivated by two main keywords: robustness and simplicity. We haven't found in the literature any branching reconstruction algorithm that could completely satisfy the combination of those two important features, which was the motivation of the hereby presented algorithm. In our algorithm the robustness is insured by the topology-driven modeling approach and the simplicity is the consequence of the use of elementary modeling tools.

III. Overview

We rely on a strong topological model, the 2-maps [27]. Our algorithm indeed takes place in the field of topologically-driven modeling. This implies that we will always separate the topology of the object from its embedding during the reconstruction; in this way we have full control of both. Also we decomposed the reconstructed model into elementary steps to guarantee an overall robustness. The first step is the reconstruction of the branches outside of the branching areas (after pre-processing). The second step is the construction of the intersections and the topological merging with the branches, leading to a connected 2-mesh. The last step smoothes the mesh using Catmull-Clark subdivision.

IV. Background

Topological model: half-edges and 2-maps A 2-dimensional mesh consists of the discretization of a surface in a cellular complex composed of cells of different dimensions (faces, edges, vertices) connected by adjacency relationships. The half-edge data structure uses the adjacencies between edges to represent the topology of the mesh of an orientable 2-manifold. Arbitrary polygonal meshes can be represented. Two types of information are needed to fully describe an object: the *topological relationships*, and the *embedding*, i.e. the geometrical data associated with each topological entity. In our case, we simply associate $3D$ points to the vertices of the mesh. The embedding of the other cells is computed by linear interpolation. The combinatorial maps were introduced in the field of topological-based modeling since [27] and [28]. These models were used in computational geometry [29], [30] and more recently to define multi-resolution models [31].

Input data The input data consist of a graph where each node stores its coordinates, local radius and connectivity information. In medical applications, the cross-sections might not be circular in reality, but are assumed as such in practice for modeling simplicity. Our test data consist of pre-processed graphs from vascular trees in the liver, provided by the medical research institute IRCAD in Strasbourg. Two main steps are necessary to obtain this graph: the first one processes the voxel data into a skeleton using Lamy's ITK filter [32], [33]. The voxel data can be quite large, especially if containing high-degree furcations. The skeleton is then modeled as a graph (indeed an oriented tree) as described in [34].

V. Our algorithm

In this section we detail each step of our algorithm. Here is an overview: after pre-processing the data, we perform a straightforward extrusion step to reconstruct the tubes outside the branching regions, followed by the main contribution of this paper, the reconstruction of the branching regions. Finally those two entities are merged into a single object which may then be subdivided if needed. We also discuss our topological model.

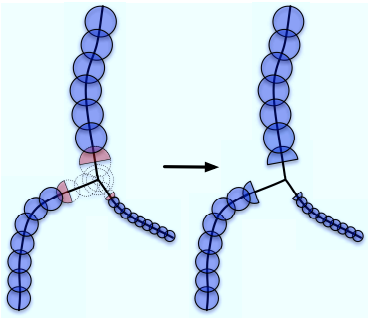


Figure 1. *Pre-processing of the branching area.* The doted circles represent the branch pieces that will be removed, the blue disks and half-disks are the remaining branch pieces, and the red half-disks illustrate our test criterion.

A. Pre-processing

We perform two pre-processing steps: the first one simplifies the tubes whereas the second one shifts the tubes' ends in the branching region.

Tube simplification Note that this simplification may only be applied to noise-free data. Our input data have already been de-noised; therefore we may apply this naive simplification. The simplification of the tubes is a classical pre-processing step: we reduce the number of spheres describing the tube's geometry. Indeed the input graph is very dense due to high data redundancy. In order to get faces as square as possible during the generation process we use the branch's radius as a basis: we start from the beginning of the branch and remove all the points located within the radius of the current point. As soon as we meet a point outside of the current radius, we iterate the process from this new point.

Branching area pre-processing We denote by *branching area* an intersection region, i.e. a region where two or more branches intersect. This second pre-processing is of great importance: we shorten the tube's ends in the branching area to avoid geometric intersections. Indeed the branches can overlap in the branching region and thus create undesirable inner faces. This is clearly not acceptable considering our goal. We therefore removed those possible overlaps. We chose to shorten the end branches by following this criterion: *any half-sphere starting from the end of a branch must not intersect the other branches linked to the same intersection*. In other words, we make sure that the end branches do not intersect but stay as close as possible from each other in order to avoid large intersection areas that could bias the 3*D* tree model. The branching region pre-processing step is illustrated in Figure 1.

B. Extrusion

The extrusion operation is a tool that allows to create objects whose shape are described by a path along which the section doesn't vary too much, e.g. a tube. In our definition, the goal is to automatically generate a tube as regular as possible starting from a path and an initial geometry provided by the user. The extrusion separates the topology and the geometry: first, a topological extrusion is performed from the two input structures; then the embedding of the vertices is applied. Compared to a basic extrusion algorithm, we have two more degrees of freedom: the scale of the extrusion (radius variation) and the orientation of each intermediate object. Figure 2(l) sketches a basic extrusion process, without torsion and scale. It is possible to add a rotation and a scale to the shape at each step of the sweeping. Each point of the sweeping path is then associated with an angle and a scale factor. Figure 2(r) shows respectively a simple extrusion, with torsion and with different scales. The extrusion step of the algorithm ignores the branching areas; it only takes as input the pre-processed branches. In our data, a branch is represented by 3*D* points and radii. The input of the extrusion consists of a starting point (the first 3*D* point of the branch), a basis geometry (an equilateral triangle inscribed in a circle of first radius) and a path represented by a list of 4*D* points (3 for the position and 1 for the radius). For the basis object's geometry we chose an equilateral triangle in order to obtain the simplest geometry possible and ease the reconstruction of the intersections. The user may freely choose the sweeping shape, e.g. a polygon, but it is recommended to choose a triangle.

C. Intersections

Constructing the intersections is obviously the most difficult task. Our idea, yet simple, appeared to be very powerful: we consider the intersection between vessels as a projected convex hull. At first sight we could simply consider the convex hull of the end branches triangles.

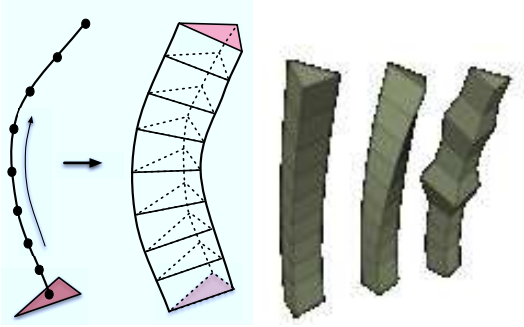


Figure 2. **Left**: *Extruding a triangle along a poly-line gives a curved prism*. **Right**: *Basic, twisted, and radius-varying extrusion*. The twist parameter can be determined by the orientation of the start and end triangles.

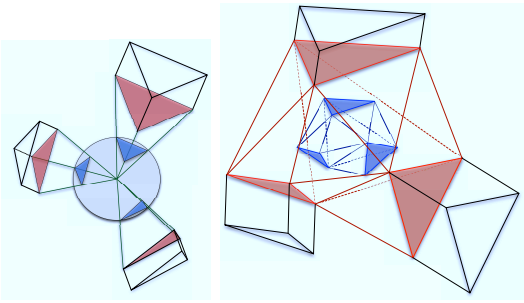


Figure 3. **Left**: *The projection of the triangles on the sphere*. In black, we see the end prisms of the tubes in the branching area; in red, the end triangles; in blue, the triangles projected on the sphere. **Right**: *The convex hull construction*. In black, we see the end prisms of the tubes in the branching area; in blue, the topological convex hull on the sphere; in red, the intersection reconstruction.

But their orientation is very arbitrary in practice and may lead to a non-convex shape. In other words, the convex hull may not contain the initial triangles; this would make the topological reconstruction tedious. Note that we only need the convex hull construction in order to be able to reconstruct the intersection but we do not necessary want a convex shape as output.

Therefore the key idea is to first project the tubes' end points of the branching area on a sphere as illustrated in Figure 3(l). This way, we first perform a robust topological reconstruction of the convex hull on the sphere. This gives us the connectivity between the tubes in the branching area. Only then will the embedding of the vertices be performed. Note that whenever we mention the topology of the reconstruction, we use, in practice, our topological library CGoGN [35]. As instance, the construction of the topological convex hull on the sphere is simply defined as a set of half-edges and their connectivities, and by a temporary embedding (before re-embedding the vertices at the final position).

The sphere is centered at the point which defines the intersection and the sphere's radius corresponds to the furthest end branch point. The projection on the sphere guarantees that all the faces' points will be on the convex hull. We conjecture that the branching area pre-processing step insures that the triangular faces will not intersect when projecting them on the sphere. This also guarantees that we will get the right faces in the convex hull in order to properly merge the prisms with the intersections. Once the points are mapped to the sphere we compute the convex hull of this set of points using a basic incremental algorithm. It takes as input a tetrahedron formed by four non coplanar points within the set of points. For each point of the set we then search, within the current convex hull, the set of faces that see the point to be added as outside of the current convex hull. Those visible faces are then deleted from the convex hull and new faces are created around the added point. We chose to apply a simple convex hull algorithm as the small number of points we are dealing with doesn't justify a more elaborated algorithm. The convex hull constructed on the sphere is shown in blue in Figure 3(r).

The last step of the intersection reconstruction is to project the coordinates of the end branches back to their original positions. The triangles and their adjacencies are kept as is. But the resulting projection might not be convex as shown in red in Figure 3(r) depending on the branching area configuration. The intersection polyhedron's mesh can be optimized by modifying the end triangles' orientation; this will change the corresponding extruded prisms by using the twist parameter. Thanks to this fully-automatic and generic reconstruction method we are able to create any kind of intersection, regardless the number of branches, their radii, etc.

Four types of intersections are illustrated in Figure 4. From left to right, we see a classical V-like intersection, the intersection between two branches, the intersection of a large and small

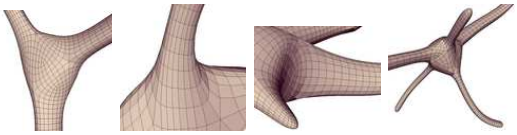


Figure 4. *Examples of (subdivided) intersections.*

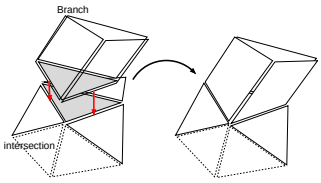


Figure 5. *The* sewVolume *operation.*

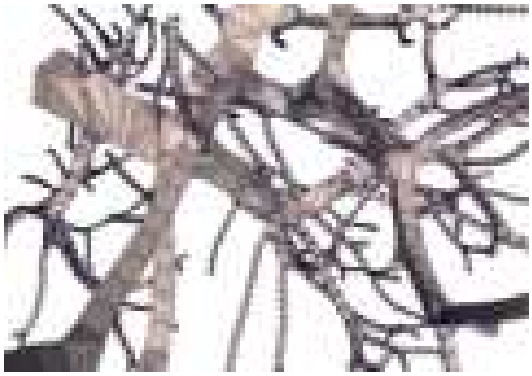


Figure 6. *The resulting base mesh and its topology.*

branch very close to each other and finally, the intersection between five branches.

D. The base mesh

Once the prisms and intersections are built we merge those two entities into a single object by traversing the object and performing a topological operation (sewVolume) which sews all the volumes together. On each polyhedron which represents an intersection, there are some triangles (the input data of the convex hull algorithm) which correspond to an extremity of the connected branches. As shown in Figure 5 the sewing algorithm consists of finding neighbors of the two triangles, and sew them together. The two original triangles are destroyed in order to merge the two volumes and keep the boundary of the surface manifold. This operation is purely topological. We do not have to care about embedding because the two triangles we use for the sewVolume operation have the same position.

Before performing this sewing operation, the mesh consists in two topological sets: the extruded branches from one side, and the intersections from the other side. Only visually seems the mesh correct as the two embeddings match each other. But the topological merge is mandatory in order to get a single mesh, especially before applying a subdivision scheme. The resulting object is a 2-manifold mesh with correct topology and compact and adaptive geometry. Note that, due to the re-embedding of the intersection points, we might obtain some (geometric) self-intersections, but the topology remains correct. Figure 6 shows the base mesh with the portal vein data set and its topology. As shown previously, our algorithm relies on a strong topological model. In practice, our implementation exploited IGG research group's common library called CGoGN [35] which provides several tools to explore the topology (or connectivity) of the reconstructed object.

E. Subdivision

The Catmull-Clark algorithm [36] is used in computer graphics to create smooth surfaces by subdivision surface modeling. After one subdivision step, the mesh will consist of quadrilaterals only. Also, it is well-known that the limit surface obtained by the refinement process is at least C^1 at extraordinary vertices and C^2 everywhere else. To get a smoother mesh we applied Catmull-Clark subdivision. Currently, for the embedding, we only use as input the base mesh vertices but in future work we will also integrate the corresponding skeleton coordinates in the subdivision scheme, in order to constraint the new positions to be as close as possible from the skeleton; this should also reduce the current bulge's effect at intersections. The number of faces increases by a factor of four at each subdivision step which leads to performance penalty. But as we control the subdivision's depth, we can choose it properly according to the available CPU. Subdivision algorithms are also being directly integrated in new GPUs and therefore only the base mesh would be required. Until this hardware integration becomes available, we have to choose to work either with the base mesh or with a smoother mesh, depending on the application and speed requirements. Note that our

topological-based implementation of Catmull-Clark will succeed even on a mesh containing intersecting triangles.

In order to preserve the overall volume of the mesh after subdivision, there are several possible ways. We may apply a correction vector (which projects the new points back to the initial mesh) after each subdivision step but doing this, the continuity scheme will be broken. Another solution is to increase the size of the tree's original radii but this leads to artifacts in the branching area which do not look realistic anymore. We could also model the branches with extruded polygons (instead of triangles) to better approach the geometry of the tubes but, by doing this, the construction of the topological convex hull for the intersections becomes less robust due to the generation of many thin faces. Finally, we may use an interpolating subdivision scheme (e.g. the Butterfly scheme [37]) which keeps in place the original vertices and avoids volume shrinking. But these algorithms may create peaks in the mesh due to the extraordinary vertices used in the subdivision scheme. Our feeling is that the most successful solution for the volume shrinking problem might be to incorporate information about the original data in the subdivision scheme.

VI. Results

We tested our algorithm on two medical data sets: the portal vein (376 vessels) and the vena cava (630 vessels) but no case-study for medical application has yet been performed. The timings for the base mesh reconstruction were respectively 6.3 and 12.0 seconds. The benchmarks were performed on a 2.5GHz Intel Core 2 Duo MacBook Pro.

VII. Conclusion

We have demonstrated a fully automatic and generic reconstruction algorithm for tubular structures which can handle arbitrary n-furcation. The resulting mesh has a consistent geometry and topology and is well-suited for medical applications. Our algorithm is based on a half-edge-like topological structure which guarantees the consistency of the topology of the reconstructed mesh. In practice it requires a topological modeling library such as CGoGN [35] which is very powerful but involves a certain expertise. But in general our algorithm is quite simple. Indeed, the other tools we used are common and easily reproducible in modeling: extrusion, convex hull, subdivision.

Limitations Our method preserves the topology of the vascular structure, but only poorly preserves the volumetric geometry, especially after subdivision. To overcome this problem we would need to post-process our model e.g. by perturbing the base mesh according to the voxel data.

Future Work There are improvements or new experimentations that could be made with our algorithm. For example we may use a more subtle branch simplification, e.g. based on the curvature that would be adaptive. Also, in the branching area pre-processing step, there might be a way of even less shortening the branches. Still concerning the construction of the intersection it would be interesting to project the points on an ellipsoid instead of a sphere; this way, there would be more degrees of freedom and thus a better fitting to the original data. Finally, as we already mentioned, we haven't yet solved the volume shrinking problem.

Acknowledgment

This work was supported by the European Commission (PASSPORT-STREP project). Thanks to Aaron Knoll for proof-reading the paper.

References

[1] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 163–169, 1987.

[2] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel, "Multi-level partition of unity implicits," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 463–470, 2003.

[3] C. Kirbas and F. Quek, "A review of vessel extraction techniques and algorithms," *ACM Comput. Surv.*, vol. 36, no. 2, pp. 81–121, 2004.

[4] "S. oeltze," IEEE Visualization 2006 Tutorials, Baltimore, MA, USA, 2006.

[5] G. Gerig, T. Koller, G. Székely, C. Brechbühler, and O. Kübler, "Symbolic description of 3-d structures applied to cerebral vessel tree obtained from mr angiography volume data," in *IPMI '93: Proceedings of the 13th International Conference on Information Processing in Medical Imaging*. London, UK: Springer-Verlag, 1993, pp. 94–111.

[6] H. H. Ehricke, K. Donner, W. Koller, and W. Straer, "Visualization of vasculature from volume data," *Computers & Graphics*, vol. 18, no. 3, pp. 395–406, 1994.

[7] H. K. Hahn, B. Preim, D. Selle, and H. O. Peitgen, "Visualization and interaction techniques for the exploration of vascular structures," in *VIS '01: Proceedings of the conference on Visualization '01*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 395–402.

[8] J. Bloomenthal, "Modeling the mighty maple," *SIGGRAPH Comput. Graph.*, vol. 19, no. 3, pp. 305–311, 1985.

[9] J. C. Hart and B. Baker, "Implicit modeling of tree surfaces," in *Proc. of Implicit Surfaces '96*, 1996, pp. 143–152.

[10] X. Jin, C.-L. Tai, J. Feng, and Q. Peng, "Convolution surfaces for line skeletons with polynomial weight distributions," *J. Graph. Tools*, vol. 6, no. 3, pp. 17–28, 2001.

[11] S. Oeltze and B. Preim, "Visualization of anatomic tree structures with convolution surfaces," in *VisSym*, 2004, pp. 311–320.

[12] J. D. Boissonnat, R. Chaine, P. Frey, G. Malandain, F. Nicoud, S. Salmon, E. Saltel, and M. Thiriet, "From medical images to computational meshes," in *Conference on Modelling and Simulation for Computer-aided Medicine and Surgery (MS4CMS'02)*, ser. ESAIM: PROC, M. Thiriet, Ed., vol. 12. European Series in Applied and Industrial Mathematics, 2002, pp. 1–7.

[13] J. D. Boissonnat and S. Oudot, "Provably good surface sampling and approximation," in *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. Aire-la-Ville, Switzerland: Eurographics Association, 2003, pp. 9–18.

[14] P. Felkel, A. Fuhrmann, A. Kanitsar, and R. Wegenkittl, "Surface reconstruction of the branching vessels for augmented reality aided surgery," in *In BIOSIGNAL 2002*, vol. 16, 2002, pp. 252–254.

[15] P. Felkel, R. Wegenkittl, and K. Buhler, "Surface models of tube trees," in *CGI '04: Proceedings of the Computer Graphics International*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 70–77.

[16] S. Ou and H. Bin, "Subdivision method to create furcating object with multibranches," *Vis. Comput.*, vol. 21, no. 3, pp. 170–187, 2005.

[17] Y. Cai, X. Ye, C. Chui, and J. H. Anderson, "Constructive algorithms of vascular network modeling for training of minimally invasive catheterization procedure," *Adv. Eng. Softw.*, vol. 34, no. 7, pp. 439–450, 2003.

[18] J. Lluch, R. Vivó, and C. Monserrat, "Modelling tree structures using a single polygonal mesh," *Graphical Models*, vol. 66, no. 2, pp. 89 – 101, 2004.

[19] X. Tian, G. Han, M. Chen, and Z. Situ, "Skeleton-based surface reconstruction for visualizing plant roots," *International Conference on Artificial Reality and Telexistence*, vol. 0, pp. 328–332, 2006.

[20] A. Bornik, B. Reitinger, and R. Beichel, "Reconstruction and representation of tubular structures using simplex meshes," in *In Proc. of Winter School of Computer Graphics (WSCG)*. Press, 2005, pp. 61–65.

[21] Y. Zhang, Y. Bazilevs, S. Goswami, R. L. Bajaj, and T. J. R. Hughes, "Hughes: Patient-specific vascular nurbs modeling for isogeometric analysis of blood flow (2006," in *in Proceedings of the 15th International Meshing Roundtable*. Springer, 2006, pp. 73–92.

[22] G. Hart, "Solid-segment sculptures," in *Proceedings of Colloquium on Mathematics and Art*, C. Brute, Ed. Springer-Verlag, 2002.

[23] E. Mandal, E. Akleman, and V. Srinivasan, "Wire modeling," in *SIGGRAPH '03: ACM SIGGRAPH 2003 Sketches & Applications*. New York, NY, USA: ACM, 2003, pp. 1–1.

[24] ——, "Column modeling," in *SIGGRAPH '04: ACM SIGGRAPH 2004 Sketches*. New York, NY, USA: ACM, 2004, p. 59.

[25] V. Srinivasan, E. Mandal, and E. Akleman, "Solidifying frames," in *Bridges: Mathematical Connections in Art, Music, and Science 2004*, 2005.

[26] G. Hart, "Sculptural forms from hyperbolic tessellations," in *Proceedings of IEEE Shape Modeling International 2008*, 2008, pp. 155–161.

[27] A. Vince, "Combinatorial maps," *J. Comb. Theory, Ser. B*, vol. 34, no. 1, pp. 1–21, 1983.

[28] J. F. Dufourd, "Algebraic map-based topological kernel for polyhedron modellers: algebraic specification and logic prototyping," in *Proc. Eurographics*. Hamburg: Elsevier, 1989, pp. 301–312.

[29] D. Cazier and J. F. Dufourd, "J.f.: A formal specification of geometric refinements," *The Visual Computer Journal, Vol*, vol. 15, pp. 279–301, 1999.

[30] J. F. Dufourd, "Design and formal proof of a new optimal image segmentation program with hypermaps," *Pattern Recogn.*, vol. 40, no. 11, pp. 2974–2993, 2007.

[31] P. Kraemer, D. Cazier, and D. Bechmann, "Extension of half-edges for the representation of multiresolution subdivision surfaces," *Vis. Comput.*, vol. 25, no. 2, pp. 149–163, 2009.

[32] L. Ibanez, W. Schroeder, L. Ng, and J. Cates, *The ITK Software Guide*, 2nd ed., Kitware, Inc. ISBN 1-930934-15-7, http://www.itk.org/ItkSoftwareGuide.pdf, 2005.

[33] J. Lamy, "Integrating digital topology in image-processing libraries," *Comput. Methods Prog. Biomed.*, vol. 85, no. 1, pp. 51–58, 2007.

[34] A. Charnoz, V. Agnus, and L. Soler, "Portal vein registration for the follow-up of hepatic tumours," in *Medical Image Computing and Computer-Assisted Intervention MICCAI 2004*, ser. LNCS, vol. 3217. Springer, 2004, pp. 878–886.

[35] F. IGG-LSIIT research group (Thery et al.), University of Strasbourg. Cgogn maps. [Online]. Available: https://iggservis.u-strasbg.fr/CGoGN

[36] E. Catmull and J. Clark, "Recursively generated b-spline surfaces on arbitrary topological meshes," pp. 183–188, 1998.

[37] N. Dyn, D. Levine, and J. A. Gregory, "A butterfly subdivision scheme for surface interpolation with tension control," *ACM Trans. Graph.*, vol. 9, no. 2, pp. 160–169, 1990.

Annexe C

Modèles non variétés

IEEE INTERNATIONAL CONFERENCE ON SHAPE MODELING AND APPLICATIONS (SMI) 2010[1]

X-maps: an efficient model for non-manifold modeling

David Cazier[1,2], Pierre Kraemer[1,2]
[1]LSIIT, UMR CNRS 7005
[2]University of Strasbourg, France

***Abstract*—A lot of representation schemes have been proposed to deal with non-manifold and mixed dimensionalities objects. A majority of those models are based on incidence graphs and although they provide efficient ways to query topological adjacencies, they suffer two major drawbacks: redundancy in the storage of topological entities and relationships, and the lack of a uniform representation of those entities that leads to the development of large sets of intricate topological operators.**

As regards to manifold meshes – and specifically triangular ones – compact and efficient models are known for twenty years. Ordered topological models like combinatorial maps or halfedges based data structures are widely studied and used.

We propose a new representation scheme – the extended maps or *X*-maps – that enhances those models to deal with non-manifold objects and mixed dimensionalities. We exhibit properties that allows an adaptive implementation of the cells and thus ensures that *X*-maps scale well in case of large surface areas or manifold pieces. We show that the storage requirements for *X*-maps is strongly reduced compared to the radial edge and similar structures and also present optimizations in case of triangular or tetrahedral non-manifold meshes.

***Keywords*—geometric modeling; combinatorial maps; non-manifold; topological model**

1. Introduction

Non-manifold boundary representations are often used in recent developments of CAD/CAM applications. They offer wider representation domains than the traditional manifold modeling schemes and provide a unified topological representation of cellular models, allowing combination of objects of mixed dimensionalities, from curves, to surfaces, to solids.

Researches in the area of non-manifold modeling pursue different goals: the design of topological representation schemes [21], [7], [16], [22], [10], [17], [6], [15], the specification of sets of operators [20], [14], [13], or the implementation of high-level functionalities [9], [3], [8]. This paper takes place in the first category and focuses on the combinatorial information needed to fully represent the structures of the modeled objects.

Usually in boundary representation, objects are subdivided or discretized in cells of different dimensions: vertices, edges, faces and volumes. A topological model describes the incidence and adjacency relationships between those cells. The challenge here is to obtain a wide representation domain, with a minimal memory cost, along with the definition of efficient topological queries and ways to traverse the structure. Most of the models found in the litterature for the representation of such meshes are based on the incidence graph of the cells and thus have to maintain and manipulate explicitly all the different topological entities.

In the case of 2-manifold surfaces, optimal models are known for a while. The Winged-edge [1] and Halfedge [19] data structures or the more formal combinatorial maps [5], [18] are widespread. Extensions have been proposed for 3D subdivisions [4] and generalized for any dimension and for the more general quasi-manifold case [11], [12]. These models share a same approach: the use of a unique topological entity, a compact representation of topological links and an ordered access to the adjacency relationships [2].

Using this same combinatorial approach, we define a new model, the extended-maps – or *X*-maps –, for the representation of non-manifold meshes that fulfills these objectives: show a low memory consumption, provide easy ways to traverse the modeled objects and allow the definition of optimal topological adjacency operators. Contrary to classical models, our model relies on

an implicit encoding of cells. We also present a dimensional adaptivity property for the cells that allows our model to be very scalable – i.e. to exhibit a small overhead compared to the cost of manifold models.

2. *X*-MAPS

The *X*-maps form an ordered topological model extending combinatorial and generalized maps. A *X*-map is defined as a 5-tuple $(B, \phi_1, \phi_2, \phi_3, \theta)$, where B is a finite set of darts – or halfedges. These darts are sewn around oriented face with ϕ_1 permutation. Oriented face are sewn together pairwise along common edges with ϕ_2 involution ($\phi_2 \circ \phi_2 = Id$) to form oriented volumes. These volumes are sewn pairwise along opposite oriented faces with ϕ_3 involution. Any dart can be sewn with θ permutation at a non manifold vertex.

In order to guarantee a consistent sewing of volumes along their common faces, an integrity constraint stating that $\phi_1 \circ \phi_3$ is an involution must hold. This means that only whole oriented faces can be sewn by ϕ_3. A key feature of combinatorial models is that these integrity constraints that ensure the modeled objects to be correct within the representation domain, are simple to express and to check.

Given a *X*-map, if θ is the identity – i.e. $\theta(x) = x$ for all darts – then the modeled object is a 3-manifold. Interpreting two ϕ_3-sewn opposite faces as a piece of surface, the modeled object can be seen as surface patches sewn along non-manifold edges. If θ and ϕ_3 are the identity, then the represented object is one or a set of orientable 2-manifold surfaces. This progression in the representation domain allows to give a flexible implementation of this model. This means that *X*-maps can be implemented with each dart holding an adaptive number of relations, which will be detailed in a following section.

Figure 1 shows a detail of an example *X*-map. The left image shows three oriented surfaces meeting on a common edge. A dangling edge is attached to one vertex and a face is linked to the other side of this edge. The top-center image shows the darts of the non-manifold edge and their ϕ_2 (green) and ϕ_3 (orange) links. θ links are shown in purple.

Important topological properties of the modeled objects like Euler's characteristic, genus or orientability are computed easily. This aspect is not detailed here and we refer the reader to similar works done on *n*G-maps [12].

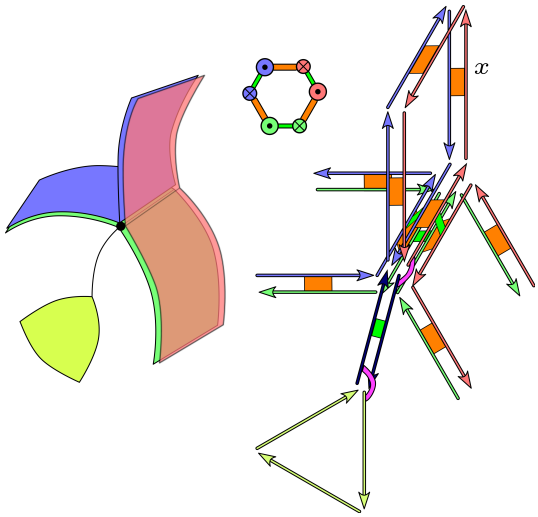


Fig. 1. Example of an *X*-map

2.1 Cells traversal

The cells of the subdivision represented by a *X*-map are not explicitly represented – as in *n*-maps or *n*-G-maps. They consist in sets of adjacent darts. Thus a cell is defined by its dimension and one of its darts. The other darts of a cell can be retrieved starting from the given dart by following a given set of relations.

For instance, starting from dart x in figure 1, the 8 darts that compose its face are retrieved following the ϕ_1 and ϕ_3 relations. This is called the orbit of x by ϕ_1 and ϕ_3 and is denoted $\langle \phi_1, \phi_3 \rangle(x)$. Volumes are the orbits of $\langle \phi_1, \phi_2 \rangle$, edges the orbits of $\langle \phi_2, \phi_3 \rangle$ and vertices the orbits of $\langle \phi_1 \circ \phi_2, \phi_1 \circ \phi_3, \theta \rangle$.

This notion of orbit provides a convenient and formal way to describe cells (and more generally paths) without any a priori knowledge about the concrete implementation of the darts and relations.

2.2 Embedding model

This kind of model aims at strongly separate the combinatorial information and the geometrical space – or any other attribute space – in which the objects are embedded. Entities that represent

explicitly the different cells of the subdivision are considered here as an optional embedding. If no information has to be attached for example to edges or volumes, no *edge* or *volume* entity has to be declared or allocated. Even so, as it is not these entities that carry the neighborhood relations, the combinatorial information representing these topological entities is still available, and these cells can still be traversed.

There are many ways to attach such an information to the cells of the subdivision. To embed a given k-cell of the subdivision, one can choose one of its darts and make it responsible for storing a link to the embedding for the whole cell. An other way would be that all the darts of a given cell share a link to the same embedding. Obviously, in a k-dimensional subdivision, as each dart is part of k different cells (one of each dimension), it can store different links to embeddings of different dimensions. If the embeddings are stored in arrays, these links can be simply indices within these arrays.

3. Implementation issues

In this section, we present how X-maps can be implemented in practice. We propose a flexible and scalable version of this implementation in which the darts of the map only store the locally needed topological relations. We also show how it can be specialized in the case of simplicial modeling.

3.1 Dynamic data structure

Usually, data structures derived from combinatorial maps models are implemented as follows. A map consists in a set of darts. Each dart contains pointers to other darts, materializing the topological relationships. Each dart usually also maintain an integer whose bits can be used during traversals or other algorithms to mark the darts using up to 32 different markers.

The cells of the subdivision are implicitly represented by sets of darts that can be traversed using the defined orbits. These orbits can easily be implemented by iterators.

A more flexible and memory aware implementation can be achieved using the previously mentionned topological property. In most cases relations ϕ_3 and θ map darts to themselves (in manifold regions and on oriented surface patches). Using an array with an adaptive size avoids storing unnecessary relations in those darts. As a result, the size of this array is different from one dart to an other according to the local topological configuration. In practice, two bits of the markers integer are used to indicate if the ϕ_3 or θ link is available in the dart, leaving 30 usable markers which is usually widely enough for most of the traversal algorithms.

3.2 Simplicial case

When dealing with simplicial representation in which every k-dimensional cell is bounded by exactly $k+1$ vertices, specialized data structures can be designed. For example, the data structure presented in [6] allows the representation of d-dimensional simplicial complexes by representing a simplified version of the incidence graph of the subdivision. This structure suffers from the classical drawback of dealing with as many different entities as the dimension of the complex.

The X-maps model can also be specialized for a usage dedicated to the representation of simplicial meshes and be less verbose than the general formulation we gave above.

Let first consider triangle meshes. Their oriented faces are made of exactly 3 darts sewn by ϕ_1. The way these 3 darts are sewn by this relation is the same for every triangle face of the mesh and thus only the ϕ_2, ϕ_3 and θ relations are relevant. To reflect this, darts can be stored within triplets which leads to a *tri-darts* object containing 3 ϕ_2 pointers ; one ϕ_3 pointer ; and 3 θ pointers. For every triangle, this implementation saves 3 ϕ_1 pointers and 2 ϕ_3 pointers – thanks to $\phi_3 \circ \phi_1$ being an involution which is verified for every X-map. Within a tri-dart, each specific dart is identified with a 2-bits id. Given a tri-dart x, its ϕ_2-sewn tri-dart through the dart of id 01 is given by $\phi_2(x,01)$. Two bits of the markers field are set in the targetted tri-dart to specify the reached dart.

The same work can be done to specialize X-maps for tetrahedral meshes. This time, the ϕ_1 and ϕ_2 relations can be saved considering 12-uplets of darts to store the topological relations around tetrahedra. Such a specialized implementation of X-maps actually brings this model close to the structure presented in [6] in terms of properties as well as memory cost, while keeping the general and formal framework presented above.

3.3 Topological operators

Contrary to classical models based on incidence graphs which needs operators to deal with all the different represented entities, the basic operators needed to manipulate *X*-maps are reduced in number. The kernel of our modeler only defines operators to sew two darts by an involution or to insert one dart in the cycle of a permutation and the corresponding destructor to remove darts from cycles.

Such a simple set of operators simplifies the check of integrity constraints. Defining more evolved operators, like the well-known Euler's ones [13], is then straightforward. They are all built as combinations of the basic operators and their implementation do not need supplementary tests or integrity checks to ensure the construction of valid objects.

Topological queries and traversals are also much simpler to implement than with an incidence graph. For example, in the Radial Edge Structure and its derivatives, there are many topological entities: shells, faces, edges, vertices, and the corresponding "uses" encoding the cycles. The stored relations are not always exhaustive and some of them must be reconstructed on the fly during the traversals.

The cellular approach leads to the development of as many adjacency operators than possible combinations of entities and relations. For ordered models, like *X*-maps, all these operators are described as simple orbits traversals. Furthermore, for all the cells in which a natural order can be defined – like vertices umbrellas, edges or faces – these traversals are done in an optimal linear time, going through exactly the needed number of darts – i.e. one for each incident or adjacent cell.

4. Storage requirements

We now give an estimation of the memory cost of *X*-maps in three different cases: 2-manifold, 3-manifold and non-manifold objects representation. Our main comparison reference is the Partial Entity Structure (PES) presented in [10] which is the current lightest structure that supports the same general representation domain than *X*-maps. In each case we add in the comparison the cost of some more specialized classically used models. We focus here on topological information and do not take into account the storage cost needed for the geometrical information which is the same for every model. When possible and under some regularity hypothesis, the number of topological entities is expressed in function of the number of top cells of the subdivision.

4.1 2-manifolds

In the case of surface manifold modeling, when representing genus 0 objects, following the Euler formula, we have: $v - e + f = 2$, with v the number of vertices, e the number of edges and f the number of faces. In the case of triangular meshes, we have $f = \frac{2}{3} \cdot e$. We deduce $e = \frac{3}{2} \cdot f$ and $v = \frac{1}{2} \cdot f + 2$.

To compute the cost of the Partial Entity Structure, we count the number of each topological entity and multiply it by its cost in term of number of pointers. In the case of manifold triangle meshes we obtain $\mathbf{38.5} \cdot \boldsymbol{f}$ **+ 18**. A direct implementation of the incidence graph would lead to a total cost of $\mathbf{15} \cdot \boldsymbol{f}$ **+ 24**. Using the simplified version presented in [6] that encodes only partial co-boundary relations, this cost is reduced to $\mathbf{9.5} \cdot \boldsymbol{f}$ **+ 2**.

When representing orientable surfaces, only one orientation of the faces of the mesh can be stored. Within *X*-maps this means that each triangular face is composed of three darts linked by ϕ_1 permutation. The triangles are linked together using ϕ_2 involution.

Within the standard implementation, our model stores $4 * 3 = 12$ pointers for each triangle, i.e. $\mathbf{12} \cdot \boldsymbol{f}$ pointers. Using the flexible version, each triangle needs $3 * 3 = 9$ pointers (each dart stores a ϕ_1 and a ϕ_2 pointer plus a pointer to the table that contains these links), which leads to a cost of $\mathbf{9} \cdot \boldsymbol{f}$.

Obviously, models dedicated to surface manifold modeling need less memory space than more general models – including ours. In the case of triangle meshes, 2-maps cost is $\mathbf{6} \cdot \boldsymbol{f}$ and 2G-maps – which can also represent non-orientable surfaces – $\mathbf{18} \cdot \boldsymbol{f}$.

4.2 3-manifolds

In the context of 3-manifold objects representation, we study the regular case of a volumetric hexahedral grid. We express here the number of stored pointers in function of the number of

hexahedra of the mesh which is here composed of the following number of cells: v volumes, $\frac{6}{2} \cdot v = 3 \cdot v$ faces, $\frac{12}{4} \cdot v = 3 \cdot v$ edges and $\frac{8}{8} \cdot v = v$ vertices.

In the PES structure the representation of this mesh needs the storage of **157** $\cdot$ $\boldsymbol{v}$ **+ 2** pointers. Using an incidence graph, the total number of pointers would be **76** $\cdot$ $\boldsymbol{v}$. The simplified version needs here only **34** $\cdot$ $\boldsymbol{v}$ pointers.

The representation of such a volumetric grid with X-maps needs 24 darts for each hexahedron which is composed of 6 oriented faces that are each composed of 4 darts linked by the ϕ_1 permutation. These 6 faces are linked together with the ϕ_2 involution. The different volumes are linked together along faces of opposite orientation using the ϕ_3 involution. With the standard implementation, our model stores $4 * 24 = 96$ pointers for each volume, i.e. **96** $\cdot$ $\boldsymbol{v}$ pointers. No memory can be saved by using the flexible version here as each dart stores a ϕ_1, a ϕ_2 and a ϕ_3 pointer plus a pointer to the table that contains these links, which makes a total of 4 pointers which is exactly the same than in the standard implementation.

Different models dedicated to volumetric manifold modeling exist. Some of them like the Facet-Edge data structure [4] or the 3G-maps model [12] (that were shown to be equivalent [11]) need more memory space than our general model. This is due to the fact that these models can also represent non-orientable volumes. In our example, these data structure cost both **128** $\cdot$ $\boldsymbol{v}$. Only 3-maps, a model dedicated to orientable volumetric meshes representation needs less memory space and cost here only **72** $\cdot$ $\boldsymbol{v}$.

4.3 Non-manifolds and mixed dimensionalities

For the case of non-manifold modeling, as there is no general formula to express the number of topological entities, we will compare the storage cost of the Partial Entity Structure and X-maps using an example mesh which is illustrated in the left part figure 2. This mesh presents some non-manifold edges as well as some non-manifold vertices. We will study the storage cost of this pattern, reproduced n times, like a town composed of buildings, each one having four antennas on the corners of its roof.

For the Partial Entity Structure, we count the number of each entity in one pattern – without forgetting to divide for the shared ones that lie on the border of the pattern – and multiply it by its cost. We obtain **489** $\cdot$ $\boldsymbol{n}$ **+ 8** stored pointers. The complete incidence graph of this object encodes **232** boundary and co-boundary links. Its simplified version encoding only partial co-boundary relations would need only **138** $\boldsymbol{n}$ pointers.

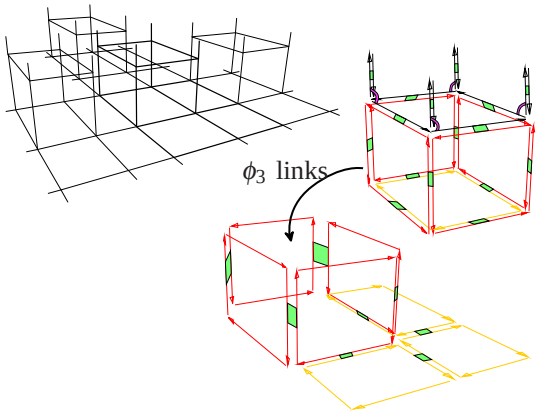


Fig. 2. Detail of the non-manifold X-map

For X-maps, we just count the number of needed darts and multiply it by the storage cost of each dart. Right part of figure 2 shows the representation of one pattern of the object in a X-map. This object can be split into two main parts: a cube that carries the four dangling edges, and a surface that presents a place to hold this cube. The purple link is a θ relation that links the dangling edges to the top vertices of the cube.

This object is composed of 60 darts, which makes a total of $60 * 4 =$ **240** $\cdot$ $\boldsymbol{n}$ stored pointers when using the standard implementation. To compute the memory cost of the flexible implementation, we have to count the number of relations maintained by each dart. The 12 darts in black maintain each a ϕ_1, a ϕ_2 and a θ relation which makes four pointers (with the pointers table pointer). The 32 darts in red maintain each a ϕ_1, a ϕ_2 and a ϕ_3 relation which also makes four pointers. Finally, the 16 darts in yellow maintain each a ϕ_1 and a ϕ_2 relation which makes three pointers. When multiplying the number of darts by their respective cost, we obtain a total cost of the structure reduced to **224** $\cdot$ $\boldsymbol{n}$ pointers.

TABLE 1
SUMMARY OF STORAGE COSTS

	2-manifold (triangle mesh)	**3-manifold** (hexahedral mesh)	**Non-manifold** (example)
Half-Edge / 2-maps	6f	–	–
2G-maps	18f	–	–
3-maps	9f	72v	–
Facet-Edge / 3G-maps	24f	192v	–
Incidence Graph	15f+24	76v	232n
Simplified Incidence Graph	9.5f+2	34v	138n
Partial Entity Structure	38.5f+18	157v+2	483n+8
X-maps	9f	96v	224n
Ratio (X-maps / PES)	∼**24%**	∼**61%**	∼**46%**

4.4 Summary

The costs of all the previously cited models in the different cases are recapitulated in the table 1. The Partial Entity Structure, which claimed to be one of the less memory space consuming structure for non-manifold modeling, needs from about **160%** to **400%** the memory space of X-maps to store the topological data.

The main reason is the use in X-maps of a unique kind of entity, namely the darts, opposing to the traditional approaches that multiply the types of entities which not only complicates the data structures but also the development of the associated operators. Moreover, the flexible implementation of X-maps proposes to save a lot of memory when dealing with objects that present only a few non-manifold cells by focusing the cost on the concerned darts. Let us recall that the incidence graph and its simplified version can be more concise, but they do not carry orientation information, and thus some of the topological traversals can not be done in optimal time.

5. CONCLUSION

We presented in this paper a new model for the representation of non-manifold meshes named X-maps. This model is expressed within the formal framework of combinatorial maps. The strong separation between the combinatorial information and the embedding space, along with the simplicity of dealing with only one basic entity leads to the development of simple and efficient operators for both the construction and the traversal of the topological structure of the objects.

The definition of the model allows a constructive approach that widens the representation domain step by step, from orientable 2-manifolds to non-manifold and mixed dimensionalities objects. This property leads to a memory efficient implementation in which the cost is focused on the concerned areas.

The benefits in terms of storage requirements are important. We have shown that our model needs only about half of the memory space used by existing data structures having a similar representation domain. X-maps also scale well to more reduced domains like simplicial complexes by exploiting a specialized implementation that makes it equivalent to existing simplicial structures.

REFERENCES

[1] B.G. Baumgart. Winged edge polyhedron represent. Technical Report CS-TR-72-320, Standford University, Department of Computer Science, October 1972.

[2] E. Brisson. Representing geometry structures in d dimensions : Topology and order. *Discrete & Computational Geometry*, pages 387–426, 1993.

[3] G.A. Croker and W.F. Reinke. An editable nonmanifold boundary representation. *IEEE Computer Graphics & Applications*, pages 39–51, 1991.

[4] D.P. Dobkin and M.J. Laszlo. Primitives for the manipulation of three-dimensional subdivisions. In *Proc. of 3rd ACM Symposium on Computional Geometry*, pages 86–99, 1987.

[5] J. Edmonds. A combinatorial representation for polyhedral surfaces. In *Notices Amer. Math. Soc.*, volume 7, 1960.

[6] L. De Floriani and A. Hui. A data structure for non-manifold simplicial d-complexes. In *Proc. of Eurographics Symposium on Geometry Processing*, pages 83–92, 2004.

[7] E.L. Gursoz, Y. Choi, and F.B. Printz. Vertex-based representation of non-manifold boundaries. *Geometric Modeling for Product Engineering*, pages 107–130, 1990.

[8] E.L. Gursoz, Y. Choi, and F.B. Printz. Boolean set operations on non-manifold boundary representation objects. *Computer-Aided Design*, 23(1):33–39, 1991.

[9] S. H. Lee. Offsetting operations in non-manifold geometric modeling. In *Proc. of 5th ACM Symposium on Solid Modeling and Applications*, pages 42–53, 1999.

[10] S. H. Lee and K. Lee. Partial entity structure: A compact non-manifold boundary representation based on partial topological entities. In *Proc. Symposium on Solid Modeling and Applications*, pages 159–170, 2001.

[11] P. Lienhardt. Topological models for boundary representation: a comparison with n-dimensional generalized maps. *Computer-Aided Design*, 23(1):59–82, 1991.

[12] P. Lienhardt. N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *International Journal on Computational Geometry and Applications*, 4(3):275–324, 1994.

[13] Y. Luo. Generalized euler operators for non-manifold boundary solid modeling. In *Geometric Modelling Studies 1990/3*, pages 19–34. MTA SZTAKI, Hungary, 1993.

[14] H. Masuda. Topological operators and boolean operations for complex-based nonmanifold geometric models. *Computer-Aided Design*, 25(2):119–129, 1992.

[15] S. Pesco, G. Tavares, and H. Lopes. A stratification approach for modeling two-dimensional cell complexes. *Computers & Graphics*, 28(2):235–247, 2004.

[16] J.R. Rossignac and M.A O'Connor. SGC: A dimension-independent model for pointsets with internal structures and incomplete boundary. *Computer-Aided Design*, 1991.

[17] V. Shapiro. Solid modeling. In G. Farin, J. Hoschek, and M.S. Kim, editors, *Handbook of computer aided geometric design*, pages 473–518. Elsevier Science Publishers, Amsterdam, 2002.

[18] A. Vince. Combinatorial maps. *Journal of Combinatorial Theory*, pages 1–21, 1983.

[19] K. Weiler. Edge-based data structures for modeling in curved-surface environments. *Computer Graphics and Applications*, 5(1):21–40, 1985.

[20] K.J. Weiler. Boundary graphs operators for non-manifold geometric modeling topology representations. *Geometric Modeling for CAD Applications*, pages 37–66, 1988.

[21] K.J. Weiler. The radial edge structure: a topological representation for non-manifold geometric modeling. *Geometric Modeling for CAD Applications*, pages 3–36, 1988.

[22] Y. Yamaguchi and F. Kimura. Nonmanifold topology based on coupling entities. *IEEE Computer Graphics and Applications*, 15(1):42–50, 1995.